Orange Pi 3
用户手册



目录

1. Orange Pi 3 的基本特性........1
1.1. 什么是 Orange Pi 3........1
1.2. Orange Pi 3 的用途........1
1.3. Orange Pi 3 是为谁设计的........1
1.4. Orange Pi 3 的硬件特性........2
1.5. Orange Pi 3 的顶层视图和底层视图........3
1.6. Orange Pi 3 的接口详情图........4
1.6.1. 带 8GB EMMC 版本........4
1.6.2. 不带 8GB EMMC 版本........5
2. 开发板使用介绍........6
2.1. 准备需要的配件........6
2.2. 下载开发板的镜像和相关的资料........8
2.3. 使用 eMMC 中预装的安卓镜像测试开发板的功能........9
2.4. 基于 Windows PC 将 Linux 镜像烧写到 TF 卡的方法........9
2.5. 基于 Ubuntu PC 将 Linux 镜像烧写到 TF 卡的方法........11
2.6. 烧写 Linux 镜像到 eMMC 中的方法........14
2.7. 烧写 Android 固件到 TF 卡的方法........14
2.8. 烧写 Android 固件到 eMMC 中的方法........17
2.9. 启动香橙派开发板........21
2.10. 调试串口的使用方法........22
2.10.1. 调试串口的连接说明........22
2.10.2. Ubuntu 平台调试串口的使用方法........23
2.10.3. Windows 平台调试串口的使用方法........26
3. Linux 系统使用说明........29
3.1. 已支持的 linux 发行版类型和内核版本........29

3.2. linux4.9 内核驱动适配情况......29
3.3. linux5.4 内核驱动适配情况......30
3.4. linux 系统默认登录账号和密码......31
3.5. Linux 桌面版系统自动登录说明......31
3.6. 第一次启动自动扩容 TF 卡中的 rootfs......33
3.7. 修改 linux 日志级别（loglevel）的方法......34
3.8. 以太网口测试......35
3.9. SSH 远程登录开发板......36
3.9.1. Ubuntu 下 SSH 远程登录开发板......36
3.9.2. Windows 下 SSH 远程登录开发板......37
3.10. HDMI 显示测试......38
3.11. Linux4.9 系统 HDMI 分辨率设置......38
3.12. Linux4.9 系统 Framebuffer 宽度和高度的修改方法......40
3.13. Framebuffer 光标相关的设置......43
3.14. 香橙派 5 寸 TFT 液晶屏测试......44
3.15. WIFI 连接测试......45
3.15.1. 服务器版镜像通过命令连接 WIFI......45
3.15.2. 服务器版镜像通过图形化方式连接 WIFI......47
3.15.3. 桌面版镜像的测试方法......50
3.16. 蓝牙使用方法......52
3.16.1. 桌面版镜像的测试方法......52
3.16.2. 服务器版镜像的使用方法......56
3.17. 板载 LED 灯测试说明......59
3.18. USB 接口测试......60
3.18.1. 连接鼠标或键盘测试......60
3.18.2. 连接 USB 存储设备测试......60
3.19. USB 无线网卡测试......61
3.20. USB 网卡测试......64

3.21. USB 摄像头测试......65
3.22. linux4.9 系统 OTG 功能测试......68
3.22.1. OTG 功能说明......68
3.22.2. OTG 接口设置为 USB Host 模式的使用方法......69
3.22.3. OTG 虚拟 USB 网卡测试......70
3.22.4. OTG 虚拟串口测试......72
3.22.5. OTG 虚拟 U 盘......74
3.23. 音频测试......76
3.23.1. linux4.9 系统耳机接口播放音频测试......76
3.23.2. linux4.9 系统 MIC 录音测试......76
3.23.3. HDMI 音频播放测试......77
3.24. 红外接收测试......77
3.25. 温度传感器......79
3.26. 安装 Docker 的方法......79
3.27. 26 Pin 接口引脚说明......81
3.28. 安装 wiringOP 的方法......81
3.29. 26pin 接口 GPIO、I2C、UART、SPI 测试......82
3.29.1. 26pin GPIO 口测试......82
3.29.2. 26pin SPI 测试......84
3.29.3. 26pin I2C 测试......85
3.29.4. 26pin 的 UART 测试......87
3.30. I2C 接口的 0.96 寸 OLED 模块使用方法......89
3.31. SPI LCD 显示屏使用方法......91
3.31.1. 2.4 寸 SPI LCD 显示屏......91
3.31.2. 3.2 寸 RPi SPI LCD 显示屏......95
3.31.3. 3.5 寸 SPI LCD 显示屏......100
3.32. 将内核打印信息输出到 26pin 串口的方法......104
3.33. 硬件看门狗测试......105
3.34. 查看 H6 芯片的 chipid......106
3.35. 烧写 linux 镜像到 eMMC 的方法......106

3.36. 开机和关机方法......109
4. Linux SDK 使用说明......110
4.1. 获取 linux sdk 的源码......110
4.1.1. 从 github 下载 orangepi-build......110
4.1.2. 下载交叉编译工具链......111
4.1.3. orangepi-build 完整目录结构说明......112
4.1.4. 从百度云盘下载 orangepi-build......113
4.2. 编译 u-boot......115
4.3. 编译 linux 内核......119
4.4. 编译 rootfs......125
4.5. 编译 linux 镜像......128
5. Android 系统使用说明......130
5.1. 已支持的 Android 版本......130
5.2. Android 7.0 功能适配情况......130
5.3. 板载 LED 灯显示说明......131
5.4. ADB 的使用方法......131
5.4.1. 打开 USB debugging 选项......131
5.4.2. 使用网络连接 adb 调试......132
5.4.3. 使用数据线连接 adb 调试......133
5.5. HDMI 4K 显示说明......134
5.6. USB 摄像头使用方法......135
5.7. Android 系统 ROOT 说明......137
6. Android SDK 使用说明......139
6.1. 下载 Android SDK 的源码......139
6.2. 搭建 Android 编译环境......139
6.3. 编译 Android 镜像......140
6.3.1. 编译内核......140
6.3.2. 编译 Android 源码......142

1. Orange Pi 3 的基本特性

1.1. 什么是 Orange Pi 3

香橙派是一款开源的单板卡片电脑，新一代的arm64开发板，它可以运行Android 7.0、Ubuntu 和 Debian 等操作系统。香橙派开发板（Orange Pi 3）使用全志 H6 系统级芯片，同时拥有 1GB/2GB LPDDR3 内存

1.2. Orange Pi 3 的用途

我们可以用它搭建：

- 一台计算机
- 一个无线网络服务器
- 游戏机
- 音乐播放器
- 高清视频播放器
- 扬声器
- Android

当然还有其他更多的功能，因为Orange Pi 是开源的

1.3. Orange Pi 3 是为谁设计的

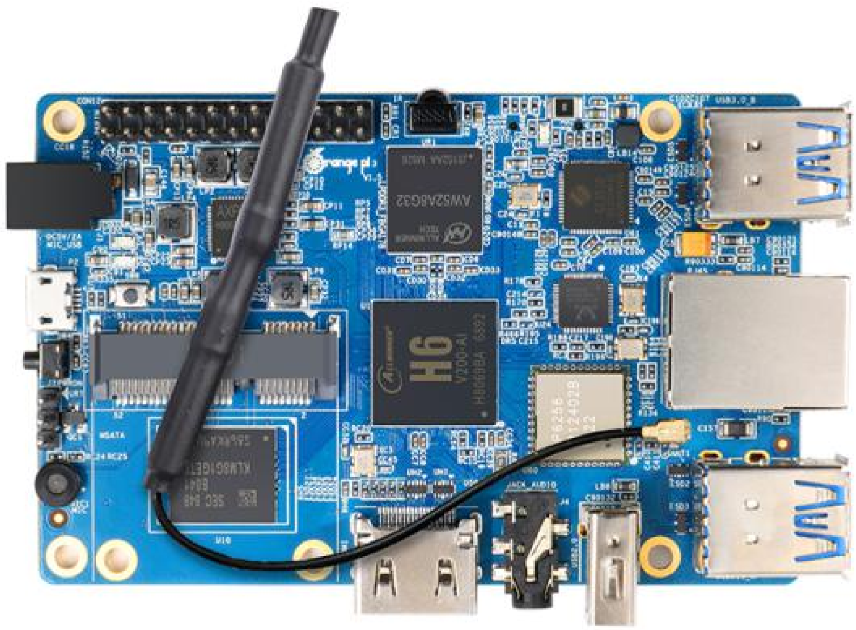
Orange Pi 开发板不仅仅是一款消费品，同时也是给任何想用技术来进行创作创新的人设计的。它是一款非常简单、有趣、实用的工具，你可以用它去打造你身边的世界

1.4. Orange Pi 3 的硬件特性

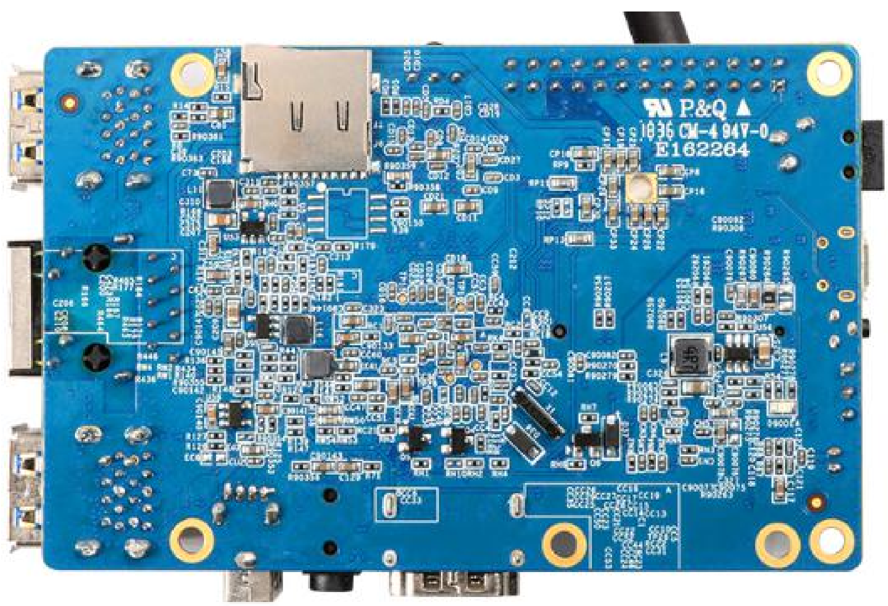
硬件特性介绍	
CPU	全志 H6 四核 64 位 1.8GHz 高性能 Cortex-A53 处理器
GPU	• High-performance multi-core GPU Mali T720 • OpenGL ES3.1/3.0/2.0/1.1
内存	1GB/2GB LPDDR3 (与 GPU 共享）
板载存储	TF 卡插槽、8GB EMMC（空贴 EMMC 的版本没有）
以太网	RTL8211E 芯片，支持 10/100M/1000M 以太网
WIFI+蓝牙	• AP6256 芯片、支持 IEEE 802.11 a/b/g/n/ac、BT5.0
mini PCIE	• Supports x1 Gen2(5.0Gbps) lane • Complies with PCI Express Base 2.0 Specification
视频输出	HDMI 2.0a 、TV CVBS output
音频输出	HDMI 输出、3.5mm 音频口
电源	DC 电源、Micro USB 接口输入
电源管理芯片	AXP805
USB 端口	4 个 USB 3.0 HOST、1 个 USB 2.0 HOST、1 个 OTG 接口
26pin 接头	带有 I2Cx1、SPIx1、UARTx1 以及多个 GPIO 口
调试串口	UART-TX、UART-RX 以及 GND
LED 灯	电源指示灯和状态指示灯
红外接收	支持红外遥控器
按键	电源按键（SW4）
支持的操作系统	Android7.0、Ubuntu、Debian 等操作系统
外观规格介绍	
产品尺寸	90mm×64mm
重量	75g
range Pi™ 是深圳市迅龙软件有限公司的注册商标	

1.5. Orange Pi 3 的顶层视图和底层视图

顶层视图：



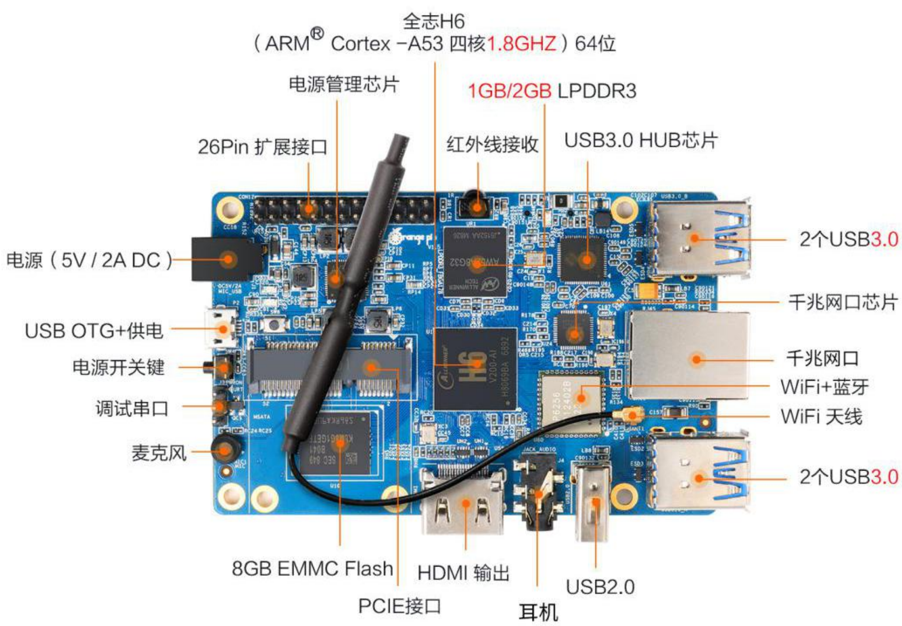
底层视图：



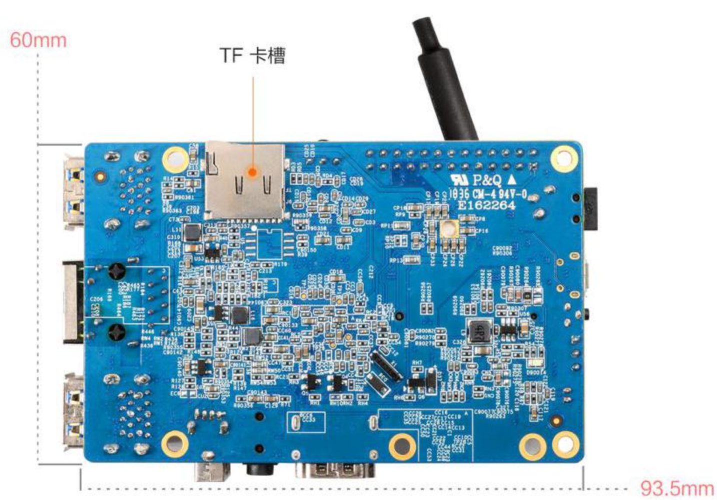
1.6. Orange Pi 3 的接口详情图

1.6.1. 带 8GB EMMC 版本

顶层视图

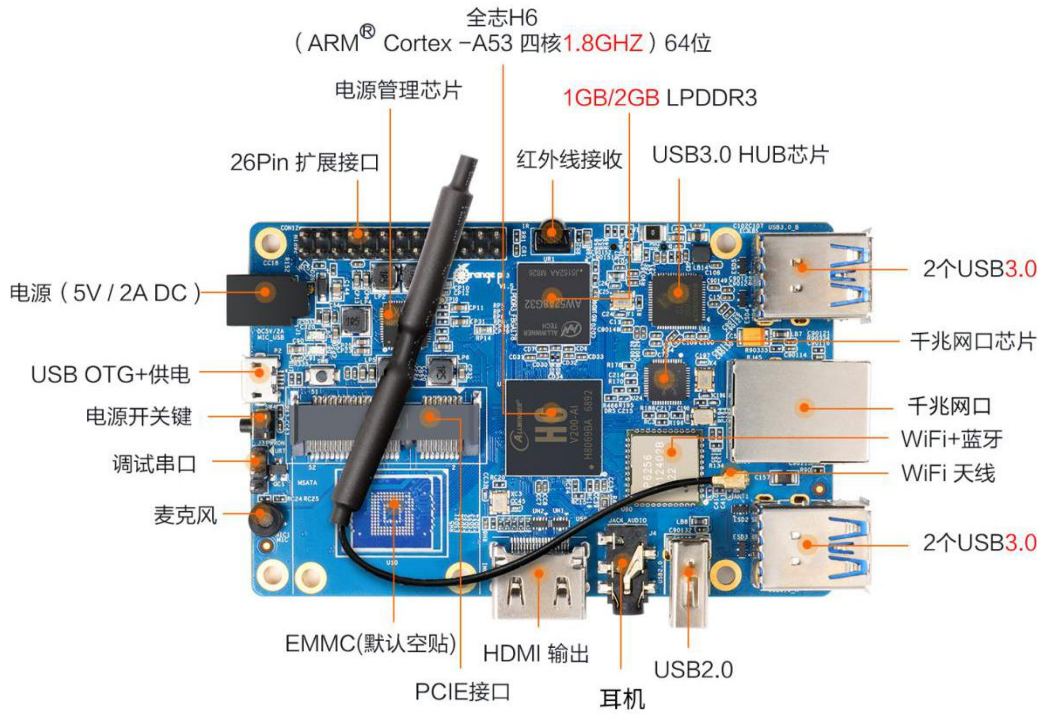


底层视图

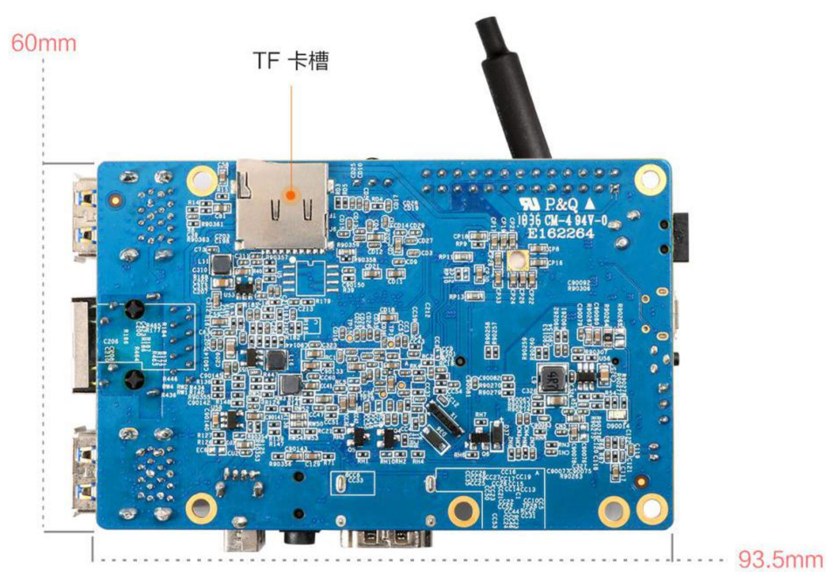


1.6.2. 不带 8GB EMMC 版本

顶层视图



底层视图



2. 开发板使用介绍

2.1. 准备需要的配件

1) TF 卡，最小 8GB 容量的 **class10** 级以上的高速卡，建议使用闪迪的 TF 卡，Orange Pi 测试都是使用闪迪的 TF 卡，其他牌子的 TF 卡可能会出现系统无法启动的问题



2) TF 卡读卡器，用于读写 TF 卡



3) HDMI 转 HDMI 连接线，用于将开发板连接到 HDMI 显示器或者电视进行显示



4) 电源适配器，5V/2A 或者 5V/3A 的高品质电源适配器，开发板的 Micro USB 接口也可以用于电源输入





5) USB接口的鼠标和键盘，只要是标准USB接口的鼠标和键盘都可以，鼠标和键盘可以用来控制Orange Pi开发板

6) 红外遥控器，主要用于控制安卓系统



7) 百兆或者千兆网线，用于将开发板连接到因特网

8) AV 视频线，如果希望通过 CVBS 接口而不是 HDMI 接口来显示视频，那么就需要通过 AV 视频线将开发板连接到电视



9) USB 转 TTL 模块和杜邦线，使用串口调试功能时，需要 USB 转 TTL 模块和杜邦线来连接开发板和电脑





10) 安装有 Ubuntu 和 Windows 操作系统的个人电脑

1	Ubuntu14.04 PC	可选，用于编译 Android 源码
2	Ubuntu18.04 PC	可选，用于编译 Linux 源码
3	Windows PC	用于烧录 Android 和 Linux 镜像

2.2. 下载开发板的镜像和相关的资料

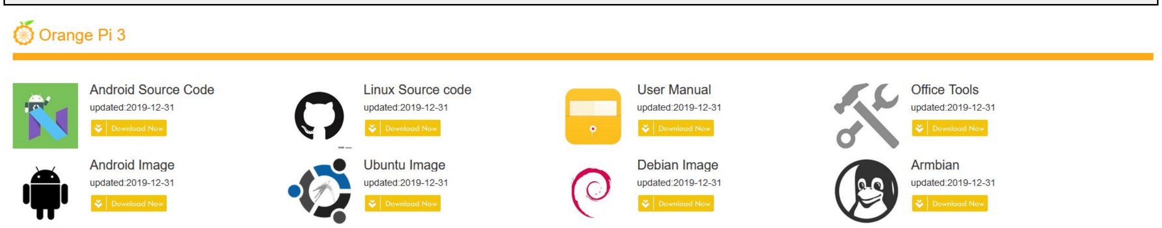
1) 中文版资料的下载网址为

```
http://www.orangepi.cn/downloadresourcescn/
```



2) 英文版资料的下载网址为

```
http://www.orangepi.org/downloadresources/
```



3) 资料主要包含

a. **Android 源码**：保存在百度云盘和谷歌网盘上

b. **Linux 源码**：保存在 github 上，链接地址为

```
https://github.com/orangepi-xunlong/orangepi-build
```

c. **用户手册和原理图：**芯片相关的数据手册也会放在这里
d. **官方工具：**主要包括开发板使用过程中需要用到的软件
e. **Android 镜像：**保存在百度云盘和谷歌网盘上
f. **Ubuntu 镜像：**保存在百度云盘和谷歌网盘上
g. **Debian 镜像：**保存在百度云盘和谷歌网盘上
h. **Armbian 镜像链接，**由 Armbian 社区开发和维护的镜像，使用过程中如果遇到问题，请先去 armbian 论坛反馈，Armbian 镜像的维护者以及其他使用 Armbian 镜像的人会协助解决各种问题，这是解决问题最快的方法。Orange Pi 不负责维护这个镜像

2.3. 使用 eMMC 中预装的安卓镜像测试开发板的功能

如果购买的是带有 8GB eMMC 版本的 Orange Pi 3 开发板，拿到开发板后，可以先使用 eMMC 中预装的 Android 镜像测试开发板的功能，确定开发板所有硬件功能都没问题后，再烧录自己想使用的系统

2.4. 基于 Windows PC 将 Linux 镜像烧写到 TF 卡的方法

1) 首先准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 **class10** 以上，建议使用闪迪等品牌的 TF 卡

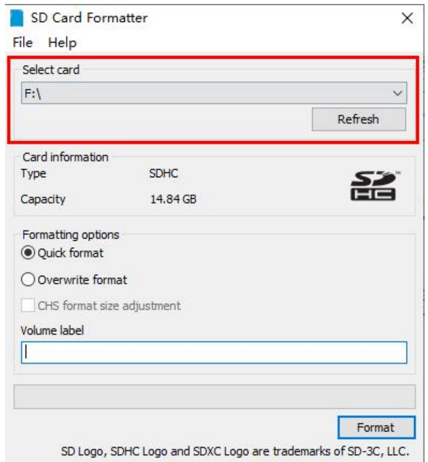
2) 然后使用读卡器把 TF 卡插入电脑

3) 接着格式化 TF 卡
   a. 可以使用 SD Card Formatter 这个软件格式化 TF 卡，其下载地址为

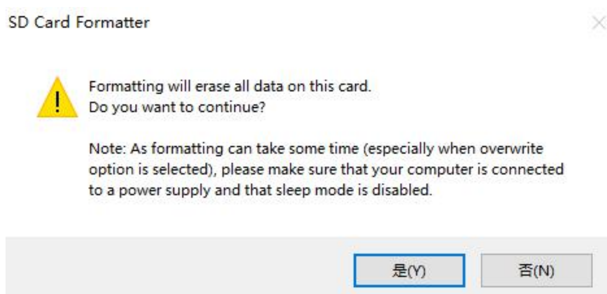
```
https://www.sdcard.org/downloads/formatter/eula_windows/SDCardFormatterv5_WinEN.zip
```

   b. 下载完后直接解压安装即可，然后打开软件

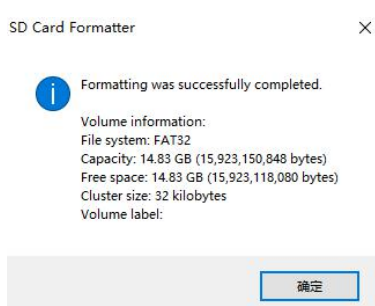
   c. 如果电脑只插入了 TF 卡，则“**Select card**”一栏中会显示 TF 卡的盘符，如果电脑插入了多个 USB 存储设备，可以通过下拉框选择 TF 卡对应的盘符



d. 然后点击“**Format**”，格式化前会弹出一个警告框，选择“**是(Y)**”后就会开始格式化



e. 格式化完 TF 卡后会弹出下图所示的信息，点击确定即可



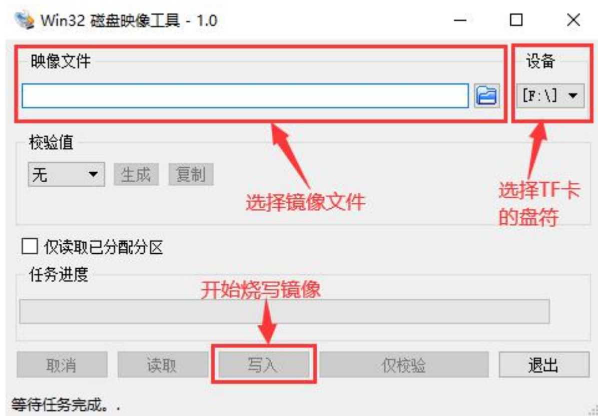
4) 从Orange Pi的资料下载页面下载想要烧录的Linux操作系统镜像文件压缩包，然后使用解压软件解压，解压后的文件中，以“.img”结尾的文件就是操作系统的镜像文件，大小一般都在 1GB以上

5) 使用 **Win32Diskimager** 烧录 Linux 镜像到 TF 卡

a. Win32Diskimager 的下载页面为

http://sourceforge.net/projects/win32diskimager/files/Archive/

b. 下载完后直接安装即可，Win32Diskimager 界面如下所示
   a) 首先选择镜像文件的路径
   b) 然后确认下 TF 卡的盘符和“**设备**”一栏中显示的一致
   c) 最后点击“**写入**”即可开始烧录



c. 镜像写入完成后，点击“**退出**”按钮退出即可，然后就可以拔出 TF 卡插到开发板中启动

2. 5. 基于 Ubuntu PC 将 Linux 镜像烧写到 TF 卡的方法

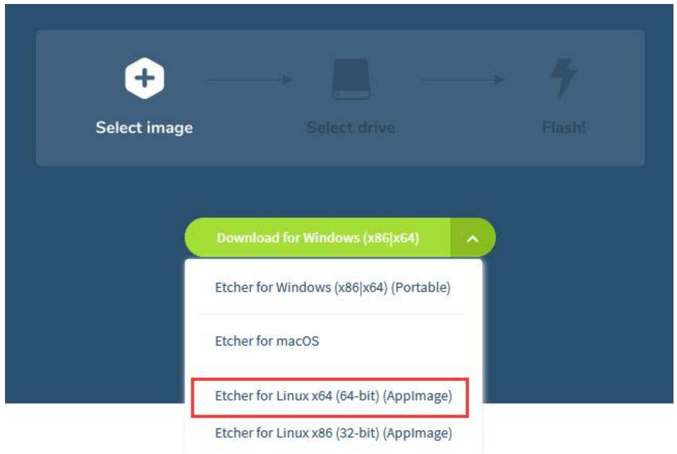
1) 首先准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 **class10** 以上，建议使用闪迪等品牌的 TF 卡

2) 然后使用读卡器把 TF 卡插入电脑

3) 下载 balenaEtcher 软件，下载地址为

https://www.balena.io/etcher/

4) 进入 balenaEtcher 下载页面后，请通过下拉框选择 Linux 版本的软件进行下载



5) 下载完后使用 **unzip** 进行解压，解压后的 **balenaEtcher-1.5.109-x64.AppImage** 就是烧录 Linux 镜像需要的软件

```
test@test:~$ unzip balena-etcher-electron-1.5.109-linux-x64.zip
Archive:   balena-etcher-electron-1.5.109-linux-x64.zip
  inflating: balenaEtcher-1.5.109-x64.AppImage
test@test:~$ ls
balenaEtcher-1.5.109-x64.AppImage    balena-etcher-electron-1.5.109-linux-x64.zip
```

6) 从 Orange Pi 的资料下载页面下载想要烧录的 Linux 操作系统镜像文件压缩包，然后使用解压软件解压，解压后的文件中，以"**.img**"结尾的文件就是操作系统的镜像文件，大小一般都在 1GB 以上

a. 7z 结尾的压缩包的解压命令如下所示

```
test@test:~$ 7z x orangepi3_2.1.0_ubuntu_bionic_server_linux4.9.118.7z
test@test:~$ ls orangepi3_2.1.0_ubuntu_bionic_server_linux4.9.118.*
orangepi3_2.1.0_ubuntu_bionic_server_linux4.9.118.7z
orangepi3_2.1.0_ubuntu_bionic_server_linux4.9.118.img.sha    #校验和文件
orangepi3_2.1.0_ubuntu_bionic_server_linux4.9.118.img     #镜像文件
```

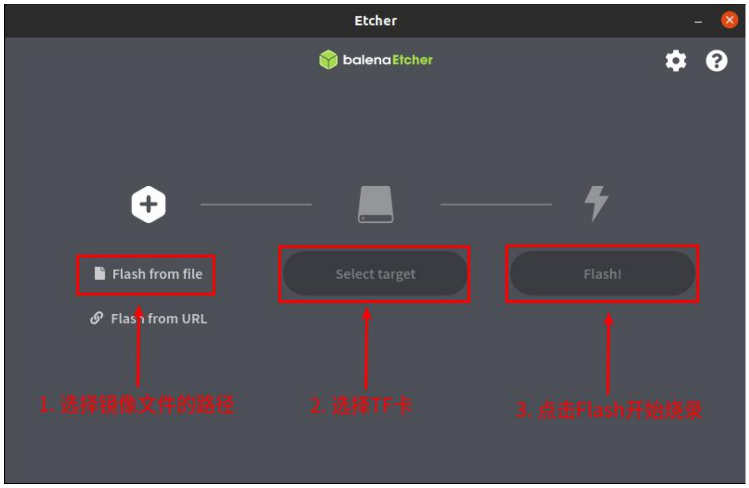
7) 解压镜像后可以先用 sha256sum -c *.sha 命令计算下校验和是否正确，如果提示**成功**说明下载的镜像没有错，可以放心的烧录到 TF 卡，如果提示**校验和不匹配**说明下载的镜像有问题，请尝试重新下载

```
test@test:~$ sha256sum -c *.sha
orangepi3_2.1.0_ubuntu_bionic_server_linux4.9.118.img: 成功
```

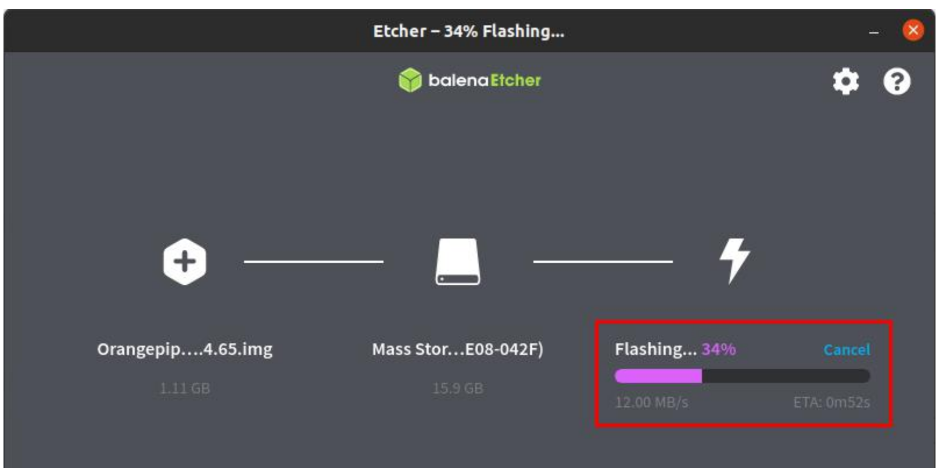
8) 然后在 Ubuntu PC 的图形界面双击 balenaEtcher-1.5.109-x64.AppImage 即可打开

balenaEtcher（无需安装），打开后的界面如下图所示

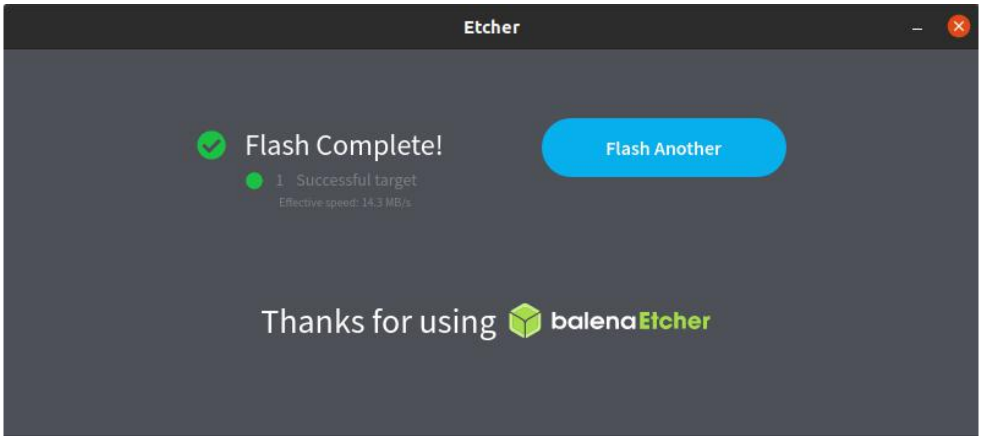
a. 首先选择 linux 镜像文件的路径
b. 然后选择 TF 卡的设备号
c. 最后点击 Flash 开始烧录



9) 烧录过程会提示写入的速度和剩余时间



10) 烧录完后会显示下面的界面，此时就可以把 TF 卡从电脑中拔出来插到开发板中启动了



2.6. 烧写 Linux 镜像到 eMMC 中的方法

见烧写 linux 镜像到 eMMC 的方法

2.7. 烧写 Android 固件到 TF 卡的方法

Android 镜像只能在 Windows 平台下使用 PhoenixCard 软件烧录到 TF 卡中，在 Linux 平台下无法烧录

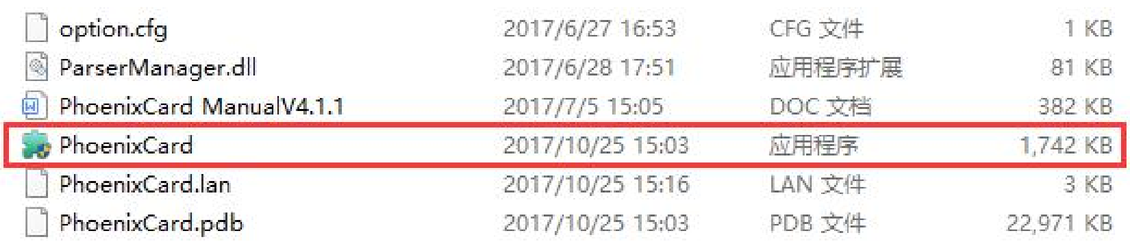
1） 首先准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 class10 以上，建议使用闪迪等品牌的 TF 卡

2) 然后使用读卡器把 TF 卡插入电脑

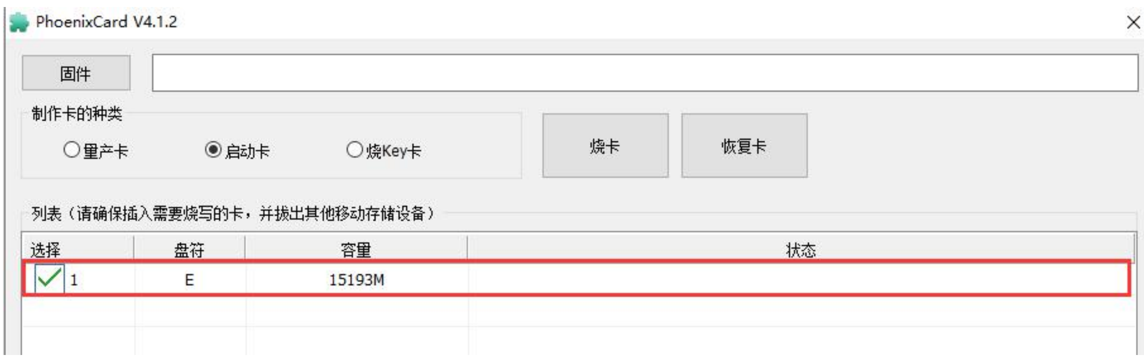
3) 从 Orange Pi 的资料下载页面下载 Android 7.0 的固件和 PhoenixCard 烧写工具，请确保 PhonenixCrad 工具的版本为 **PhoenixCard v4.1.2**

3） 使用解压软件解压下载的 Android 固件的压缩包，解压后的文件中，以“**.img**”结尾的文件就是 Android 固件

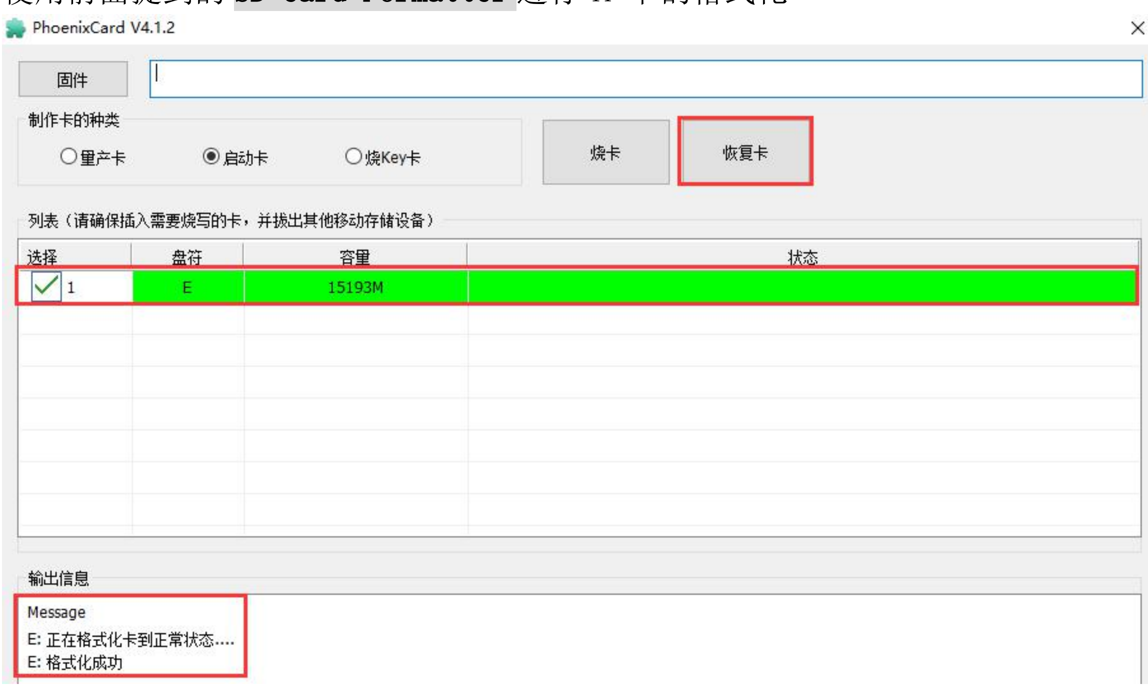
4） 使用解压软件解压 PhoenixCard v4.1.2.rar，此软件无需安装，在解压后的文件夹中找到 **PhoenixCard** 打开即可



5) 打开 **PhoenixCard** 后，如果 TF 卡识别正常，会在中间的列表中显示 TF 卡的盘符和容量，**请务必确认显示的盘符和你想烧录的 TF 卡的盘符是一致的**，如果没有显示可以尝试拔插下 TF 卡



6） 确认完盘符后，先格式化 TF 卡，点击 PhoenixCard 中**恢复卡**按钮即可，也可以使用前面提到的 SD Card Formatter 进行 TF 卡的格式化

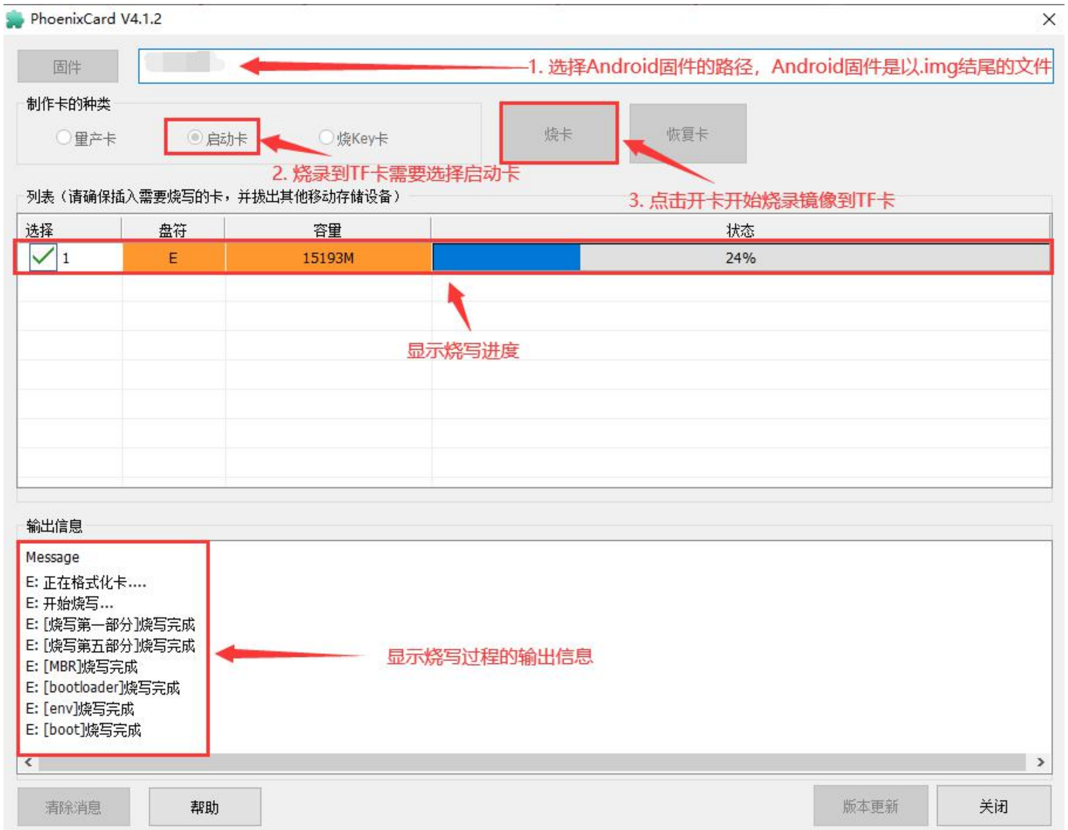


7） 然后开始将 Android 固件写入 TF 卡

a. 首先在“**固件**”一栏中选择 Android 固件的路径

b. 在“**制作卡的种类**”中选择“**启动卡**”

c. 然后点击“**烧卡**”按钮就会开始烧录



8) 烧录完后 PhoenixCard 的显示如下图所示，此时点击**关闭**按钮即可退出 PhoenixCard，然后就可以把 TF 卡从电脑中拔出来插到开发板中启动了



2.8. 烧写 Android 固件到 eMMC 中的方法

Android 镜像只能在 Windows 平台下使用 PhoenixCard 软件烧录到 EMMC 中，在 Linux 平台下无法烧录

1) 首先请注意，将 Android 固件烧录到开发板的 eMMC 中需要借助 TF 卡来完成，主要分为如下两步

a. 先使用 PhoenixCard 将 Android 固件以**量产卡**的方式烧录到 TF 卡中
b. 然后使用 TF 卡将 Android 固件烧录到 eMMC 中

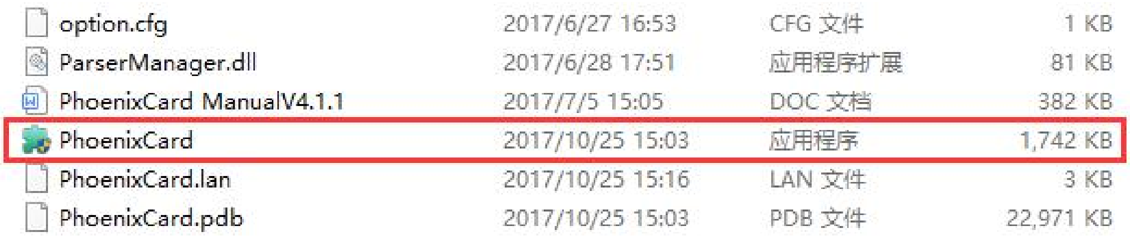
2) 先准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 class10 以上，建议使用闪迪等品牌的 TF 卡

3) 然后使用读卡器把 TF 卡插入电脑

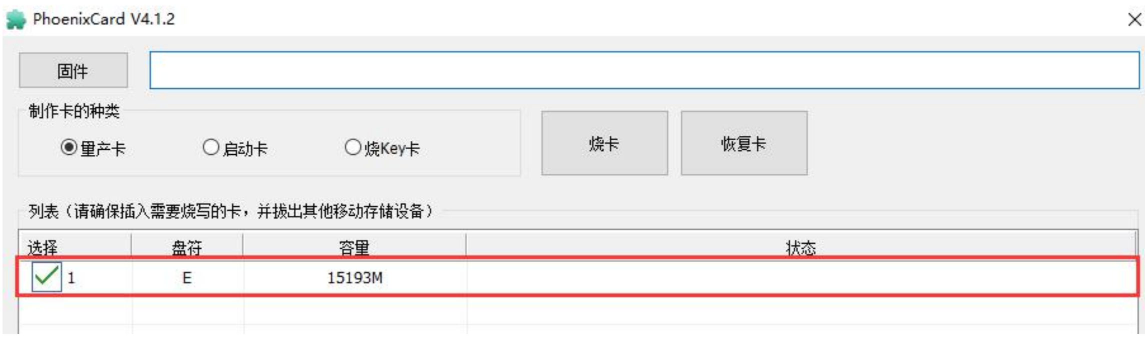
4) 从 Orange Pi 的资料下载页面下载 Android 7.0 的固件和 PhoenixCard 烧写工具，请确保 PhonenixCrad 工具的版本为 **PhoenixCard v4.1.2**

5) 使用解压软件解压下载的 Android 固件的压缩包，解压后的文件中，以 "**.img**" 结尾的文件就是 Android 固件

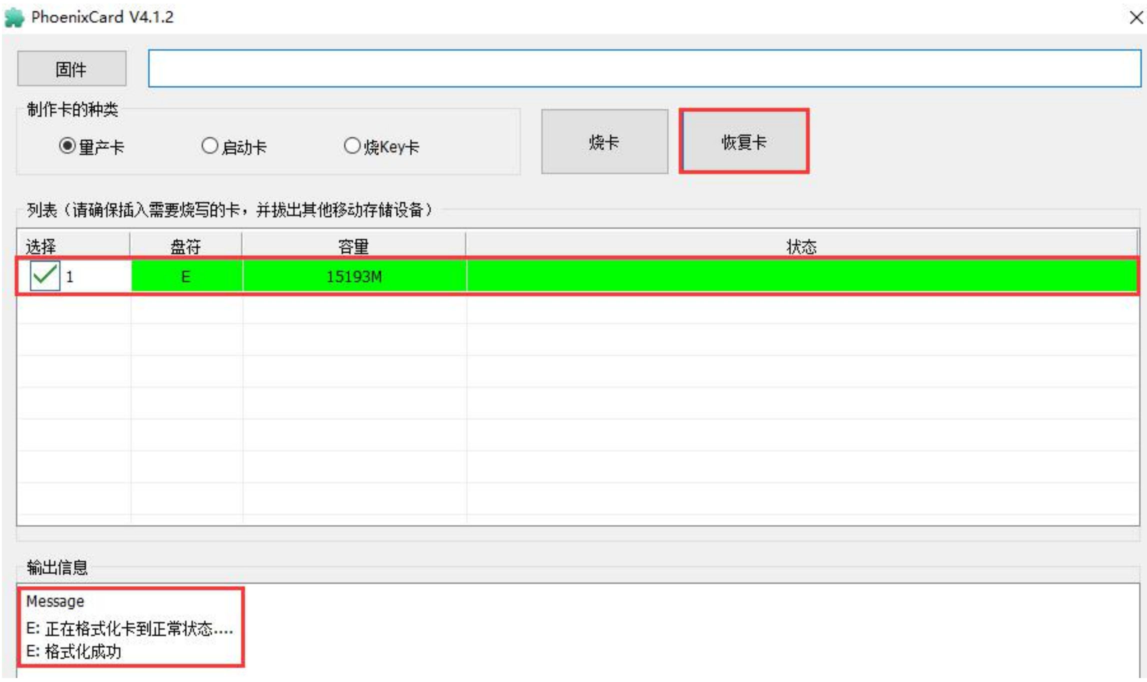
6） 使用解压软件解压 PhoenixCard v4.1.2.rar，此软件无需安装，在解压后的文件夹中找到 **PhoenixCard** 打开即可



7) 打开 **PhoenixCard** 后，如果 TF 卡识别正常，会在中间的列表中显示 TF 卡的盘符和容量，**请务必确认显示的盘符和你想烧录的 TF 卡的盘符是一致的**，如果没有显示可以尝试拔插下 TF 卡



8） 确认完盘符后，先格式化 **TF** 卡，点击 PhoenixCard 中**恢复卡**按钮即可，也可以使用前面提到的 SD Card Formatter 进行 TF 卡的格式化

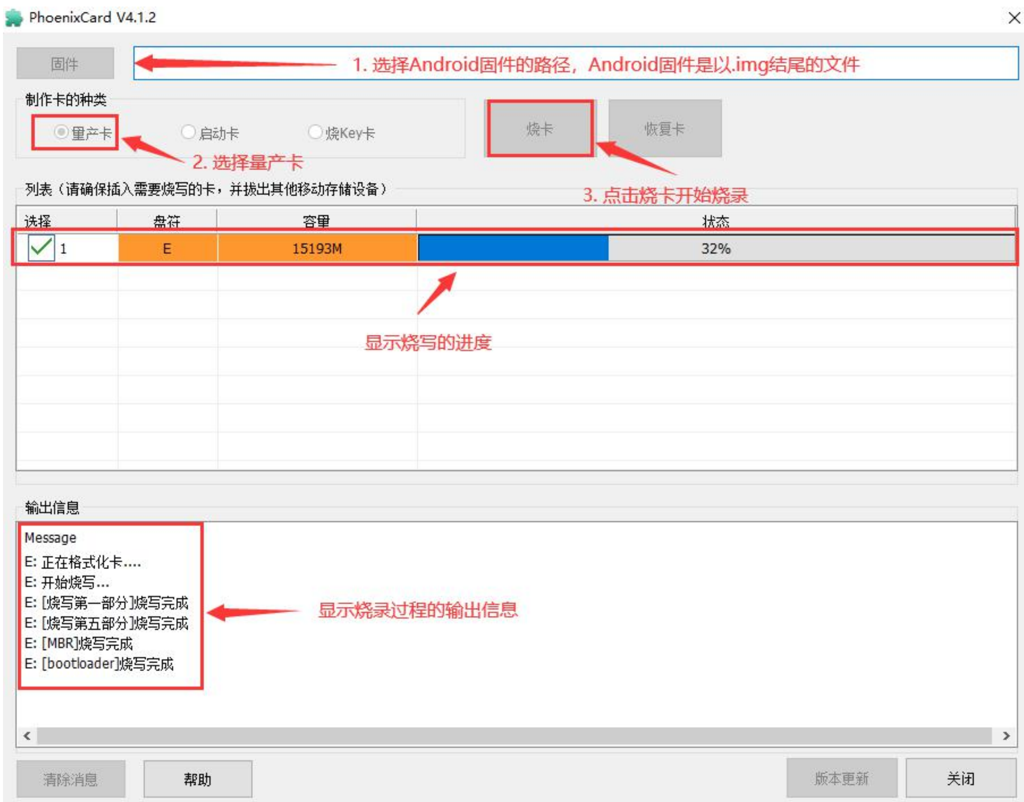


9） 然后开始将 Android 固件写入 TF 卡

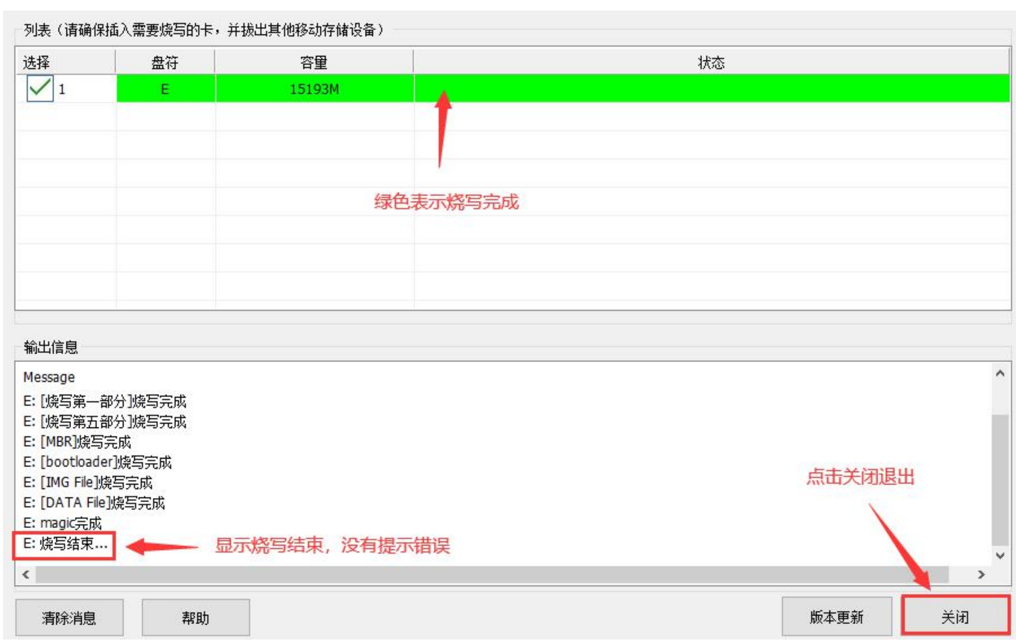
a. 首先“**固件**”一栏中选择 Android 固件的路径

b. 在“**制作**”**卡的种类**中选择“**量产卡**”

c. 然后点击“**烧卡**”按钮就会开始烧录

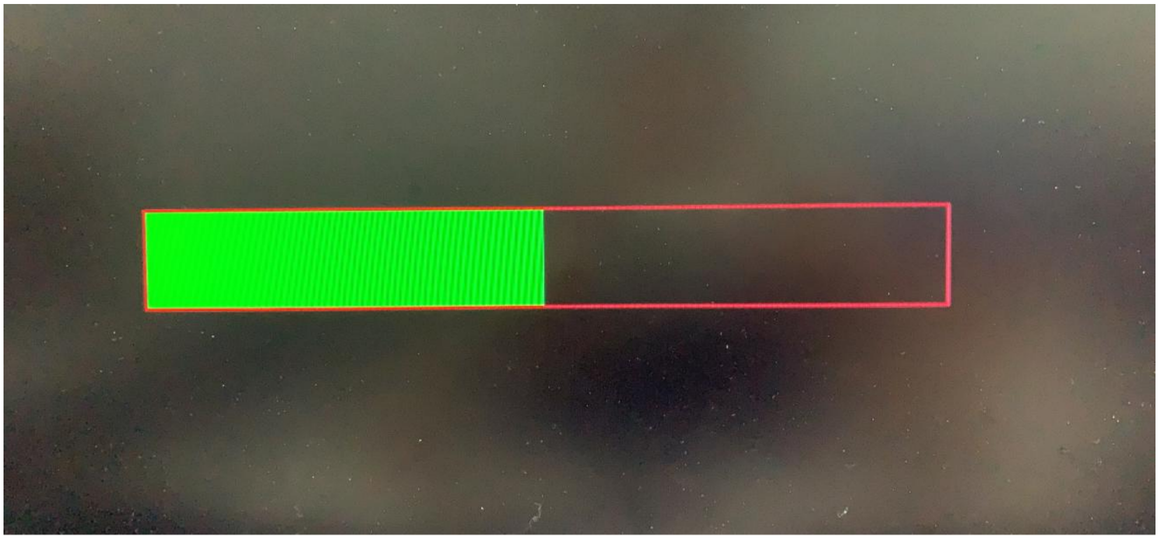


10) 烧录完后 PhoenixCard 的显示如下图所示，此时点击**关闭**按钮即可退出 PhoenixCard

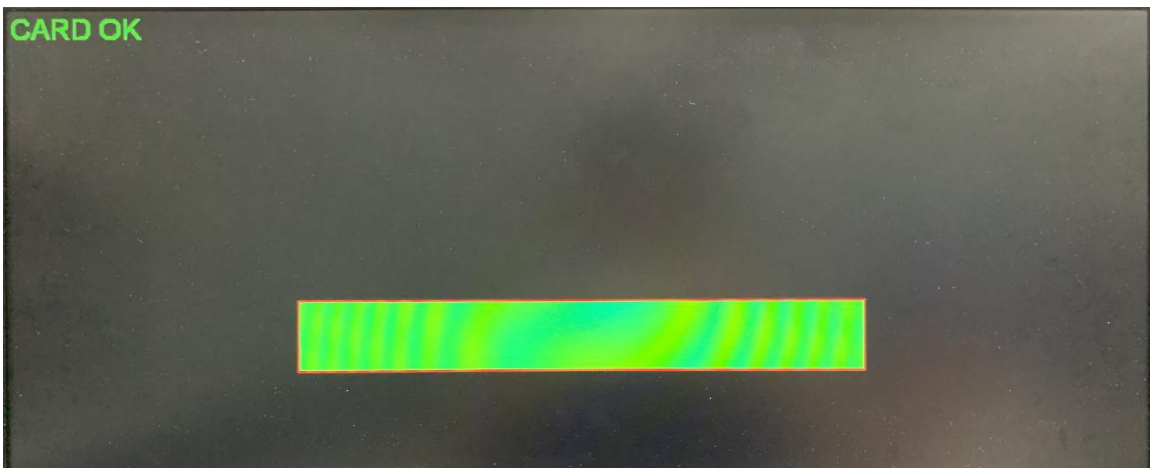


11) 然后将 TF 卡插入开发板，上电启动开发板后会自动将 TF 卡中的 Android 固件

烧录到开发板的 eMMC 中，如果开发板连接了 HDMI 显示器，在 HDMI 显示器上会显示烧录 Android 固件到 eMMC 的进度条



12) 烧录完成后 HDMI 的显示如下图所示，然后开发板会自动关机



13) 此时就可以拔出 TF 卡，然后重新上电，就会启动 eMMC 中的 Android 系统了

2.9. 启动香橙派开发板

1) 将烧录好镜像的 TF 卡插入香橙派开发板的 TF 卡插槽中
2) 开发板有 HDMI 接口，可以通过 HDMI 转 HDMI 连接线把开发板连接到电视或者 HDMI 显示器
3) 接上 USB 鼠标和键盘，用于控制香橙派开发板
4) 开发板有以太网口，可以插入网线用来上网

5) 连接一个 5V/2A（5V/3A 的也可以）**高品质**的电源适配

a. 切记不要插入 12V 的电源适配器，如果插入了 12V 的电源适配器，会烧坏开发板

b. 系统上电启动过程中很多不稳定的现象基本都是供电有问题导致的，所以一个靠谱的电源适配器很重要

6) 然后打开电源适配器的开关，如果一切正常，此时 HDMI 显示器就能看到系统的启动画面了

7) 如果想通过调试串口查看系统的输出信息，请使用 USB 转 TTL 模块和杜邦线将开发板连接到电脑，串口的连接方法请参看调试串口的使用方法一节

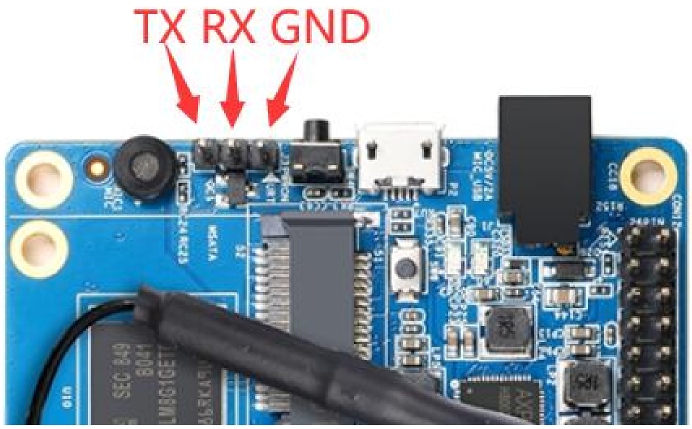
2.10. 调试串口的使用方法

2.10.1. 调试串口的连接说明

1) 首先需要准备一个 USB 转 TTL 模块，此模块在 Orange Pi 的店铺中可以买到，如果有其他类似的 USB 转 TTL 模块也可以，然后将 USB 转 TTL 模块的 USB 接口一端插入到电脑的 USB 接口中



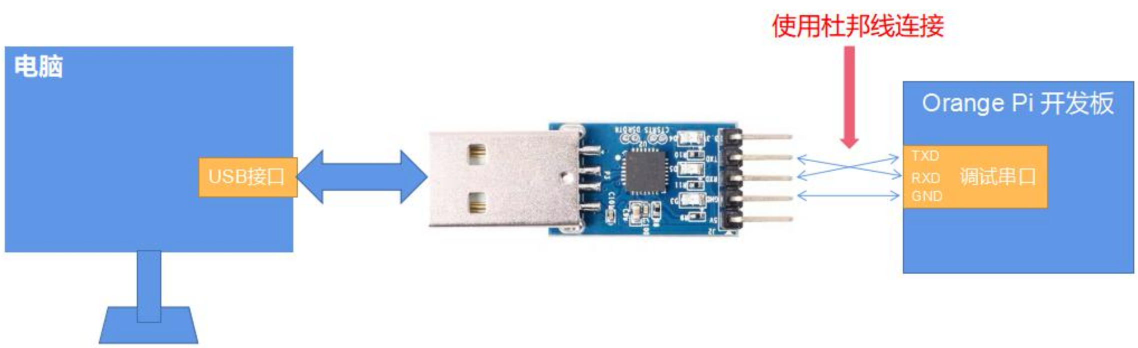
2) 开发板的调试串口 GND、TX 和 RX 引脚的对应关系如下图所示



3) USB 转 TTL 模块 GND、TX 和 RX 引脚需要通过杜邦线连接到开发板的调试串口上

a. USB 转 TTL 模块的 GND 接到开发板的 GND 上
b. USB 转 TTL 模块的 **RX 接到开发板的 TX 上**
c. USB 转 TTL 模块的 **TX 接到开发板的 RX 上**

4) USB 转 TTL 模块连接电脑和 Orange Pi 开发板的示意图如下所示



USB转TTL模块连接电脑和 Orange Pi 开发板的示意图

2. 10. 2. Ubuntu 平台调试串口的使用方法

1) 如果 USB 转 TTL 模块连接正常，在 Ubuntu PC 的/dev 下就可以看到对应的设备节点名，记住这个节点名，后面设置串口软件时会用到

```
test@test:~$ ls /dev/ttyUSB*
/dev/ttyUSB0
```

2) linux 下可以使用的串口调试软件有很多，如 putty、minicom 等，下面演示 putty 的使用方法

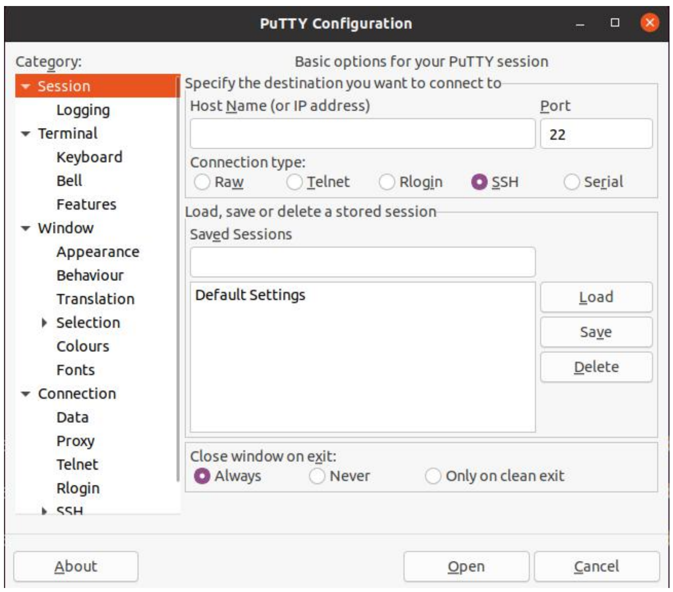
3) 首先在 Ubuntu PC 上安装 putty

```
test@test:~$ sudo apt update
test@test:~$ sudo apt install putty
```

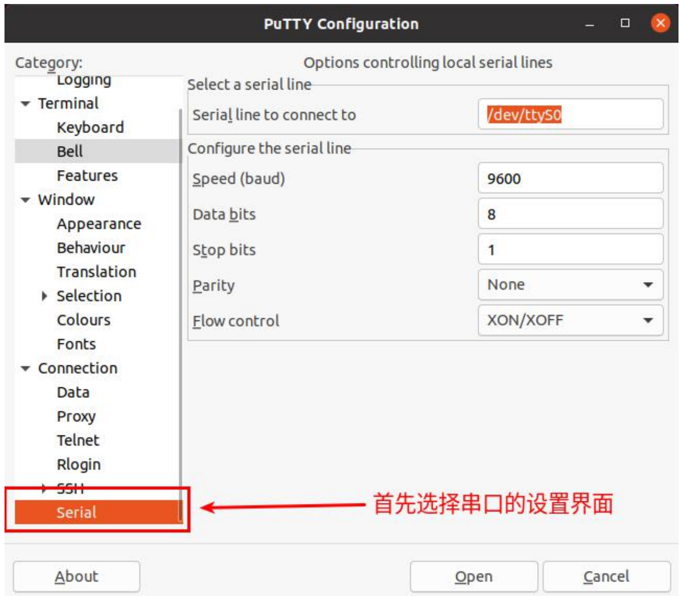
4) 然后运行 putty，记得加 sudo 权限

```
test@test:~$ sudo putty
```

5) 执行 putty 命令后会弹出下面的界面



6) 首先选择串口的设置界面

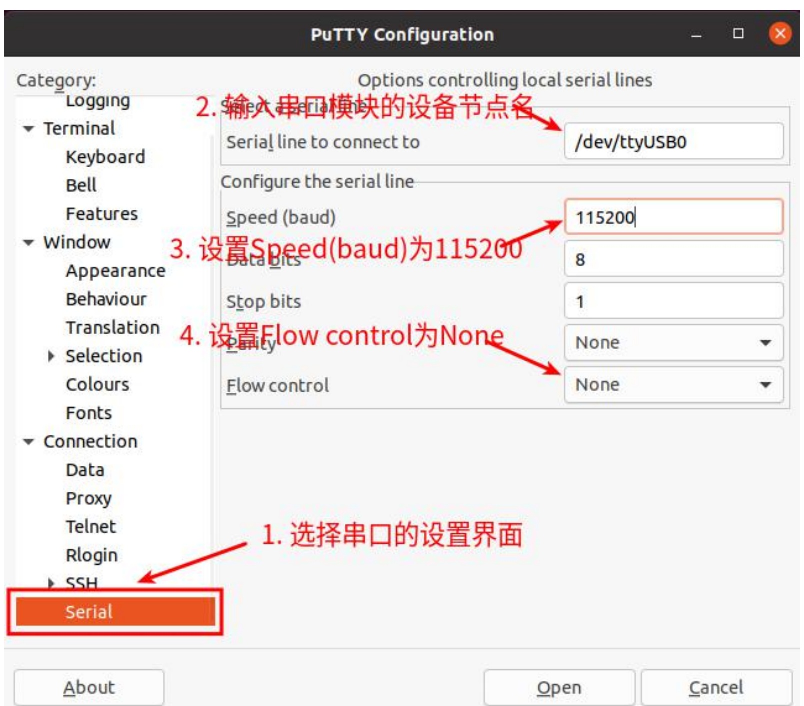


7) 然后设置串口的参数

a. 设置 Serial line to connect to 为/dev/ttyUSB0（修改为对应的节点名，一般为/dev/ttyUSB0）

b. 设置 Speed(baud)为 115200（串口的波特率）

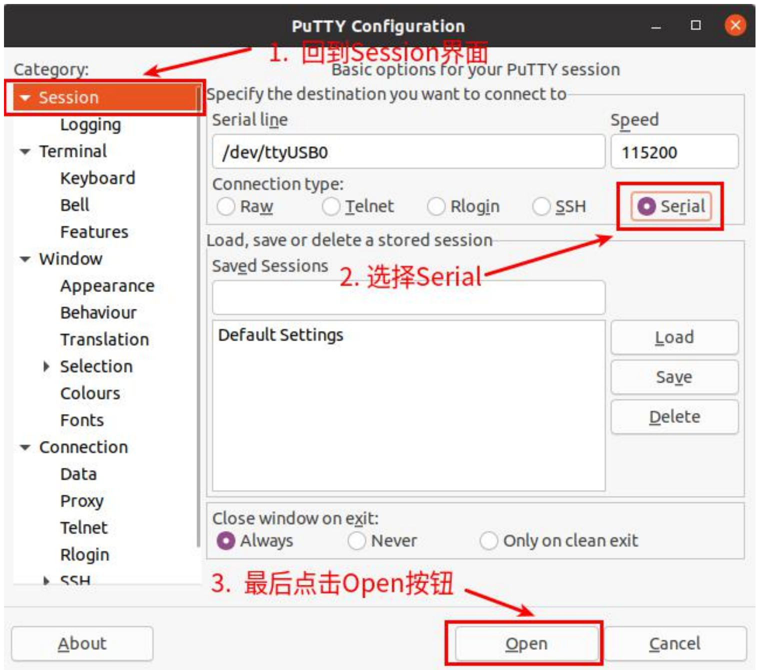
c. 设置 Flow control 为 None



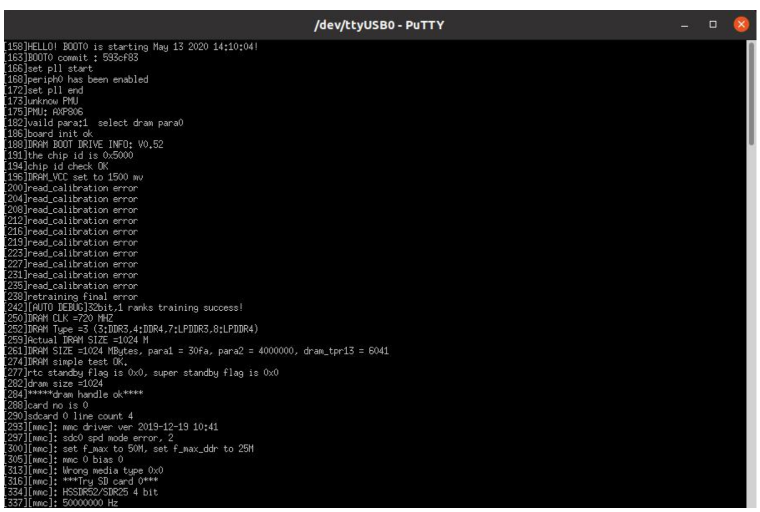
8) 在串口的设置界面设置完后，再回到 Session 界面

a. 首先选择 Connection type 为 Serial

b. 然后点击 Open 按钮连接串口



9) 启动开发板后，就能从打开的串口终端中看到系统输出的 Log 信息了



2.10.3. Windows 平台调试串口的使用方法

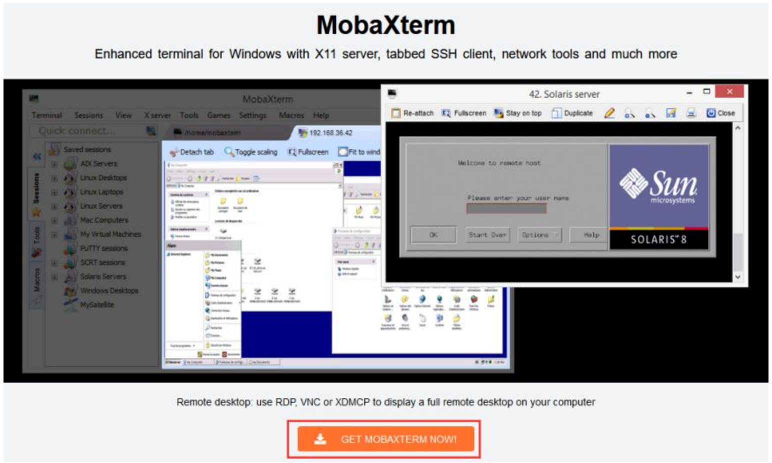
1) Windows 下可以使用的串口调试软件有很多，如 SecureCRT、MobaXterm 等，下面演示 MobaXterm 的使用方法，这款软件有免费版本，无需购买序列号即可使用

2) 下载 MobaXterm

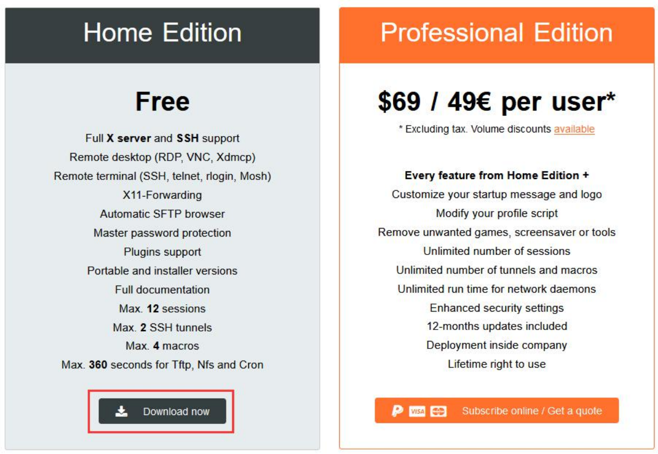
a. 下载 MobaXterm 网址如下

https://mobaxterm.mobatek.net/

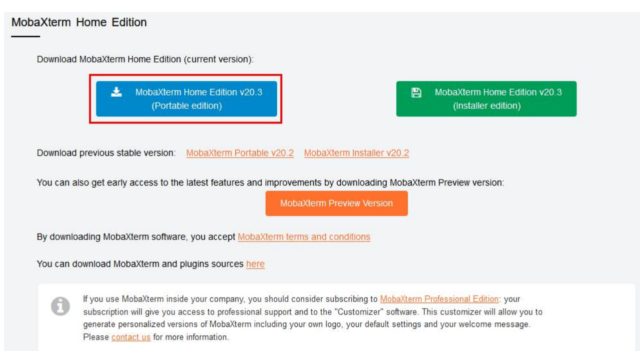
b. 进入 MobaXterm 下载网页后点击 **GET XOBATERM NOW!**



c. 然后选择下载 Home 版本



d. 然后选择 Portable 便携式版本，下载完后无需安装，直接打开就可以使用



3) 下载完后使用解压缩软件解压下载的压缩包，即可得到 MobaXterm 的可执软件，然后双击打开



4) 打开软件后，设置串口连接的步骤如下

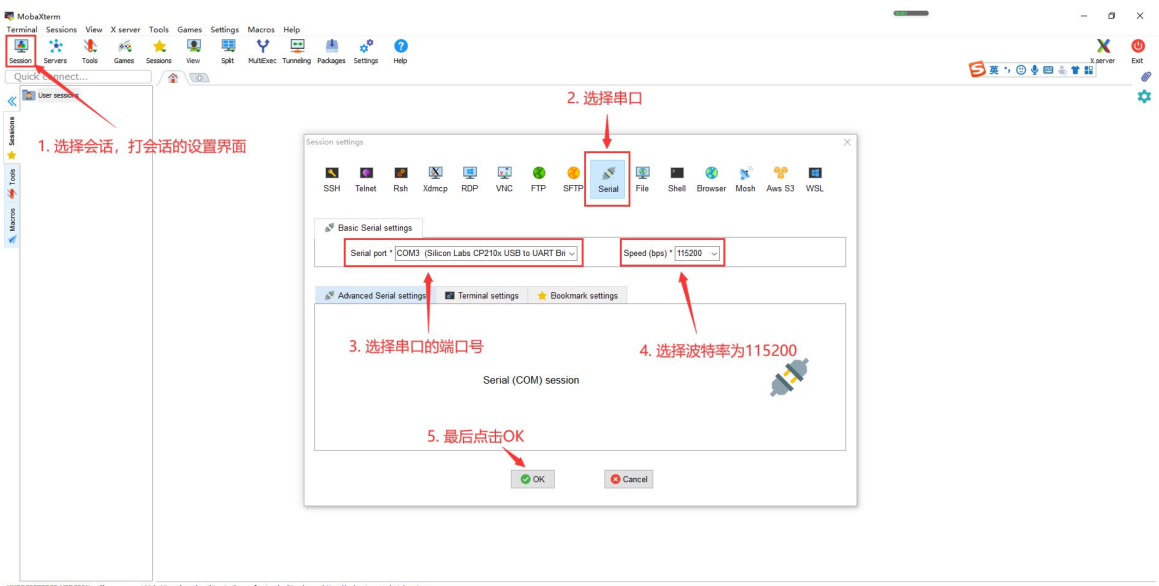
a. 打开会话的设置界面

b. 选择串口类型

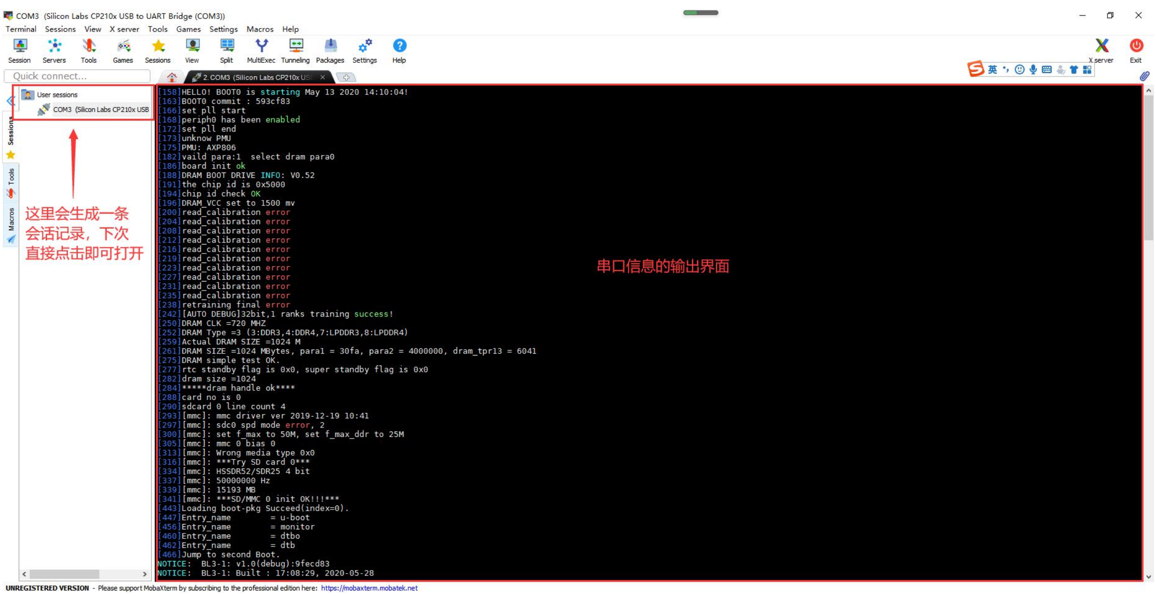
c. 选择串口的端口号（根据实际的情况选择对应的端口号），如果看不到端口号，请使用 **360 驱动大师**扫描安装 USB 转 TTL 串口芯片的驱动

d. 选择串口的波特率为 115200

e. 最后点击“**OK**”按钮完成设置



5) 点击“**OK**”按钮后会进入下面的界面，此时启动开发板就能看到串口的输出信息了



3. Linux 系统使用说明

3.1. 已支持的 linux 发行版类型和内核版本

发行版类型	内核版本	服务器版	桌面版
Ubuntu 18.04	linux4.9	支持	支持
Ubuntu 20.04	linux4.9	支持	支持
Debian 10	linux4.9	支持	支持
Ubuntu 20.04	linux5.4	支持	支持
Debian 10	linux5.4	支持	支持

3.2. linux4.9 内核驱动适配情况

功能	状态
HDMI 视频	OK
HDMI 音频	OK
USB3.0 x 4	OK
USB2.0	OK
TF 卡启动	OK

网卡	OK
红外接收	OK
WIFI	OK
蓝牙	OK
耳机音频	OK
MIC 录音	OK
USB 摄像头	OK
LED 灯	OK
26pin GPIO	OK
I2C	OK
SPI	OK
UART	OK
温度传感器	OK
硬件看门狗	OK
开关机按键	OK
EMMC 启动	OK
CVBS 视频	NO
mini PCIE	NO
Mali GPU	NO
视频编解码	NO

3. 3. linux5.4 内核驱动适配情况

功能	状态
HDMI 视频	OK
HDMI 音频	OK
USB3.0 x 4	OK
USB2.0	OK
TF 卡启动	OK
网卡	OK
红外接收	OK
WIFI	OK
蓝牙	OK
耳机音频	NO

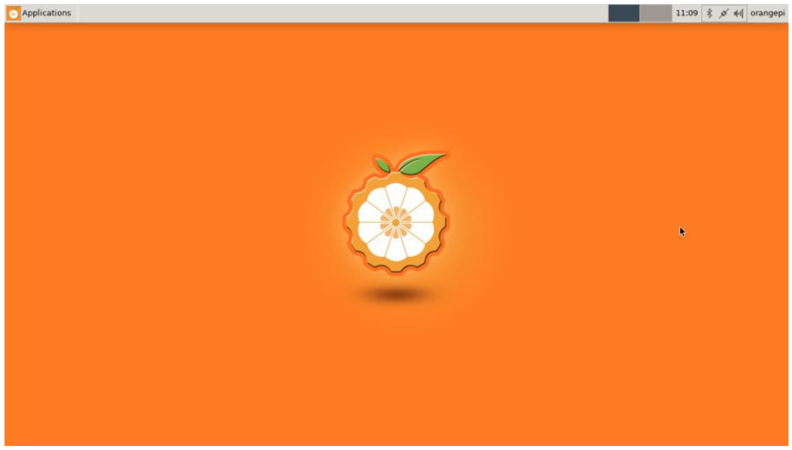
MIC 录音	NO
USB 摄像头	OK
LED 灯	OK
26pin GPIO	OK
I2C	OK
SPI	OK
UART	OK
温度传感器	OK
硬件看门狗	OK
开关机按键	OK
EMMC 启动	OK
CVBS 视频	NO
mini PCIE	NO
Mali GPU	NO
视频编解码	NO

3. 4. linux 系统默认登录账号和密码

账号	密码
root	orangepi
orangepi	orangepi

3. 5. Linux 桌面版系统自动登录说明

1) 桌面版系统默认启动后会自动登录进入桌面，无需输入密码



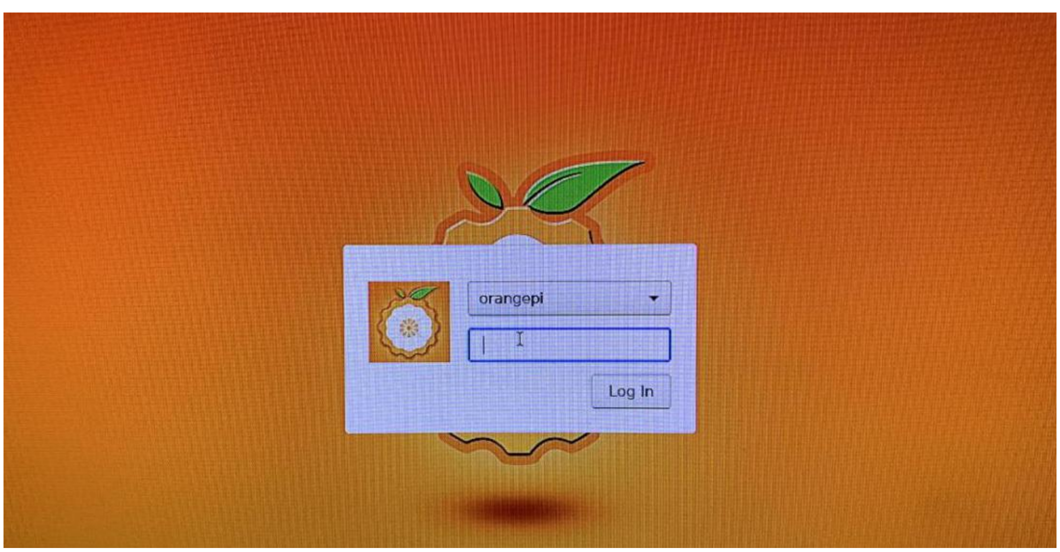
2) 修改/etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf 中的配置可以禁止桌面版系统自动登录桌面，修改命令如下，也可以打开配置文件直接修改

```
root@orangepi:~# sed -i "s/autologin-user=orangepi/#autologin-user=orangepi/" /etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf
```

3) 修改完后/etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf 的配置如下所示

```
root@orangepi:~# cat /etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf
[Seat:*]
#autologin-user=orangepi
autologin-user-timeout=0
user-session=xfce
```

4) 然后重启系统就会出现登录对话框，此时需要输入密码才能进入系统



3. 6. 第一次启动自动扩容 TF 卡中的 rootfs

1) TF 卡第一次启动 linux 系统时会通过 orangepi-resize-filesystem.service 这个 systemd 服务来调用 orangepi-resize-filesystem 脚本自动进行 rootfs 的扩容，所以**无需再手动扩容**

2) 登录系统后可以通过 df -h 命令来查看 rootfs 的大小，如果和 TF 卡的实际容量一致，说明自动扩容运行正确

```
root@orangepi:~# df -h
Filesystem          Size   Used Avail Use% Mounted on
udev                 430M       0   430M     0% /dev
tmpfs                100M   5.6M     95M     6% /run
/dev/mmcblk0p1       15G   915M     14G     7% /
tmpfs                500M       0   500M     0% /dev/shm
```

3) 需要注意的是，linux 系统只有一个 ext4 格式的分区，没有使用单独的 BOOT 分区来存放内核镜像等文件，也就不存在 BOOT 分区扩容的问题

4) 另外如果不需要自动扩容 rootfs，可以使用下面的方法来禁止

a. 首先将 linux 镜像烧录到 TF 卡中

b. 然后将 TF 卡插入 Ubuntu PC 中（Windows 不行），Ubuntu PC 一般会自动挂载 TF 卡的分区，如果自动挂载正常，使用 ls 命令可以看到下面的输出，TF 卡的分区名和下面命令所示名字不一定相同，请根据实际情况进行修改

```
test@test:~$ ls /media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db/
bin   boot   dev   etc   home   lib   lost+found   media   mnt   opt   proc   root   run
sbin   selinux   srv   sys   tmp   usr   var
```

c. 然后在 Ubuntu PC 中将当前用户切换成 root 用户

```
test@test:~$ sudo -i
[sudo] test 的密码：
root@test:~#
```

d. 然后进入 TF 卡中的 linux 系统的 root 目录下新建一个名为**.no_rootfs_resize** 的文件

```
root@test:~# cd /media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db
root@test:/media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db# cd root
root@test:/media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db/root# touch .no_rootfs_resize
```

```
root@test:/media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db/root# ls .no_rootfs*
.no_rootfs_resize
```

e. 然后就可以卸载 TF 卡，再拔出 TF 插到开发板启动，linux 系统启动时，当检测到/root 目录下有**.no_rootfs_resize** 这个文件就不会再自动扩容 rootfs 了

f. 禁止 rootfs 自动扩容后可以看到 TF 卡可用容量只有 200M 左右

```
root@orangepi:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            927M     0  927M   0% /dev
tmpfs           200M  5.6M  194M   3% /run
/dev/mmcblk0p1  1.5G  1.3G  196M  87% /
tmpfs           997M     0  997M   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           997M     0  997M   0% /sys/fs/cgroup
tmpfs           997M  4.0K  997M   1% /tmp
/dev/zram0       49M  1.5M   44M   4% /var/log
tmpfs           200M     0  200M   0% /run/user/0
```

3. 7. 修改 linux 日志级别（loglevel）的方法

1) linux 系统的 loglevel 默认设置为 1，当使用串口查看启动信息时，内核输出 log 如下所示，基本全部屏蔽了

```
Starting kernel ...

Uncompressing Linux... done, booting the kernel.

Orange Pi 2.1.0 Focal ttyS0

orangepi login:
```

2) 当 linux 系统启动出现问题时，可以使用下面的方法来修改 loglevel 的值，从而打印更多的 log 信息到串口显示，方便调试。如果 linux 系统启动失败，无法进入系统，可以把 TF 卡通过读卡器插入 Ubuntu PC 中，然后在 Ubuntu PC 中挂载 TF 卡后直接修改 TF 卡中的 linux 系统的配置，修改完后，再把 TF 卡插入开发板中启动

```
root@orangepi:~# sed -i "s/verbosity=1/verbosity=7/" /boot/orangepiEnv.txt
root@orangepi:~# sed -i "s/console=both/console=serial/" /boot/orangepiEnv.txt
```

3) 上面的命令其实都是设置/boot/orangepiEnv.txt 中的变量，设置完后可以打开/boot/orangepiEnv.txt 检查下

```
root@orangepi:~# cat /boot/orangepiEnv.txt
verbosity=7
console=serial
```

4) 然后重启开发板，内核的输出信息就都会打印到串口输出了

3.8. 以太网口测试

1) 首先将网线插入开发板的以太网接口，并确保网络是畅通的

2) 系统启动后会通过 DHCP 自动给以太网卡分配 IP 地址

3) 查看 IP 地址的命令如下

```
root@orangepi:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.47  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::e56:c34d:62f0:8d6e  prefixlen 64  scopeid 0x20<link>
        ether 02:81:3e:a8:58:d8  txqueuelen 1000  (Ethernet)
        RX packets 2165  bytes 177198 (177.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 312  bytes 40435 (40.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 39
```

4) 测试网络连通性的命令如下

```
root@orangepi:~# ping www.baidu.com -I eth0
PING www.a.shifen.com (14.215.177.38) from 192.168.1.12 eth0: 56(84) bytes of data.
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=1 ttl=56 time=6.74 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=2 ttl=56 time=6.80 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=3 ttl=56 time=6.26 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=4 ttl=56 time=7.27 ms
^C
```

```
--- www.a.shifen.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 6.260/6.770/7.275/0.373 ms
```

3. 9. SSH 远程登录开发板

Linux 系统默认都开启了 ssh 远程登录，并且允许 root 用户登录系统。ssh 登录前首先需要确保以太网或者 wifi 网络已连接，然后使用 ifconfig 命令或者通过查看路由器的方式获取开发板的 IP 地址

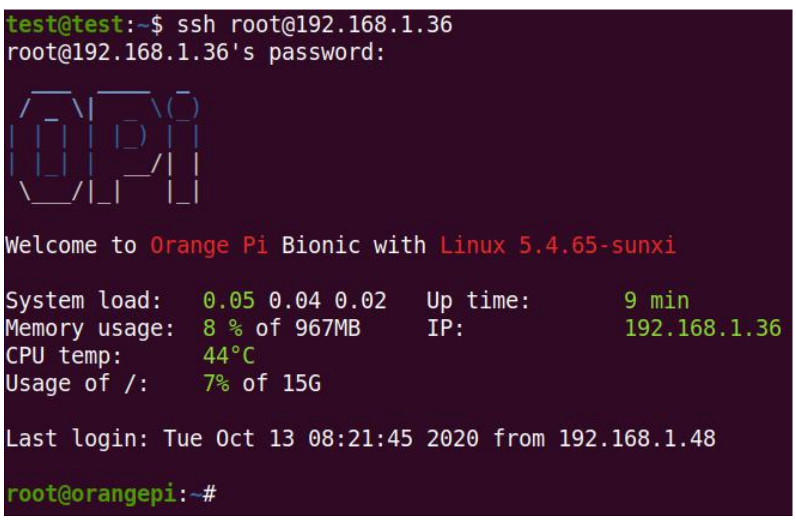
3. 9. 1. Ubuntu 下 SSH 远程登录开发板

1) 获取开发板的 IP 地址

2) 然后就可以通过 ssh 命令远程登录 linux 系统

```
test@test:~$ ssh root@192.168.1.36        (需要替换为开发板的 IP 地址)
root@192.168.1.36's password:     （在这里输入密码，默认密码为 orangepi）
```

3) 成功登录系统后的显示如下图所示



4) ssh 登录时如果提示下面的错误

```
test@test:~$ ssh root@192.168.1.36
Connection reset by 192.168.1.149 port 22
lost connection
```

可以在开发板上输入下面的命令再尝试是否能连接

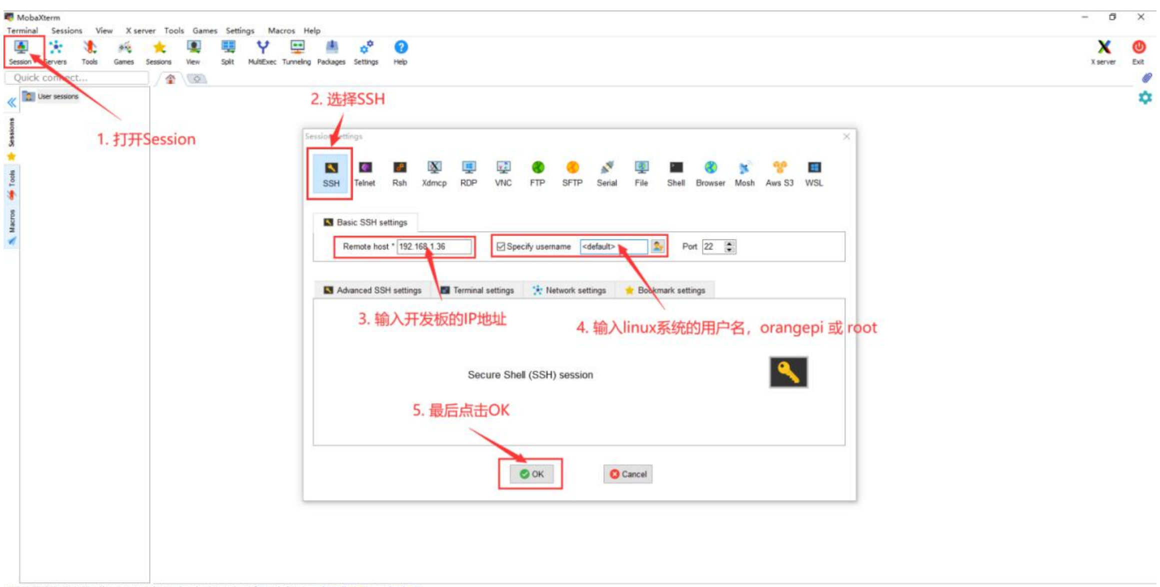
```
root@orangepi:~# rm /etc/ssh/ssh_host_*
root@orangepi:~# dpkg-reconfigure openssh-server
```

3.9.2. Windows 下 SSH 远程登录开发板

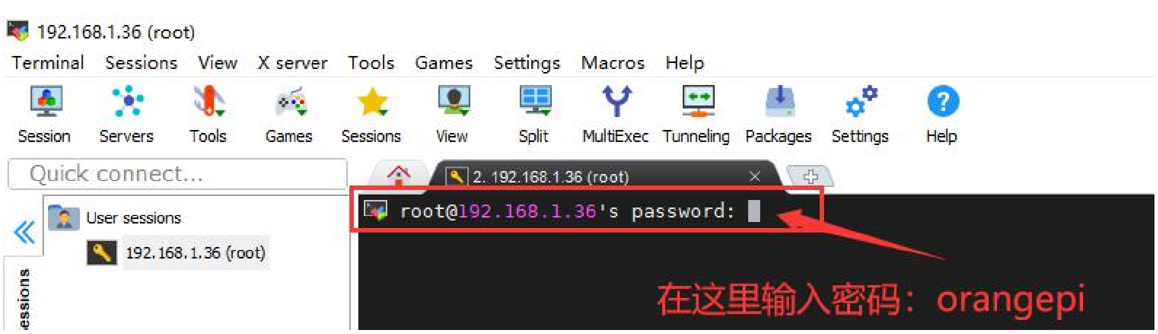
1) 首先获取开发板的 IP 地址

2) 在 windows 下可以使用 MobaXterm 远程登录开发板，首先新建一个 ssh 会话

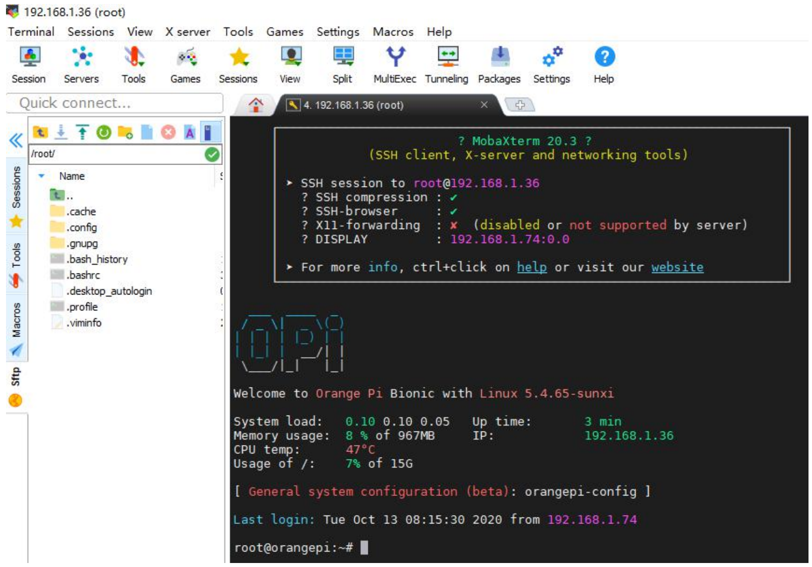
   a. 打开 **Session**
   b. 然后在 **Session Setting** 中选择 **SSH**
   c. 然后在 **Remote host** 中输入开发板的 IP 地址
   d. 然后 **Specify username** 中输入 linux 系统的用户名 **root** 或 **orangepi**
   e. 最后点击 **OK** 即可



3) 然后会提示输入密码，默认 root 和 orangepi 用户的密码都为 orangepi



4) 成功登录系统后的显示如下图所示



3.10. HDMI 显示测试

1) 使用 HDMI 转 HDMI 线连接 Orange Pi 开发板和 HDMI 显示器



2) 启动 linux 系统后如果 HDMI 显示器有图像输出说明 HDMI 接口使用正常

3.11. Linux4.9 系统 HDMI 分辨率设置

注意：此方法只适用于 linux4.9 内核的系统，linux5.4 内核的系统无法使用

1) 在 linux4.9 系统的/boot/orangepiEnv.txt 中有个 disp_mode 变量，可以通过它来设置 HDMI 输出的分辨率，linux 系统默认设置的分辨率为 1080p60

```
root@orangepi:/boot# cat orangepiEnv.txt
verbosity=1
disp_mode=1080p60
```

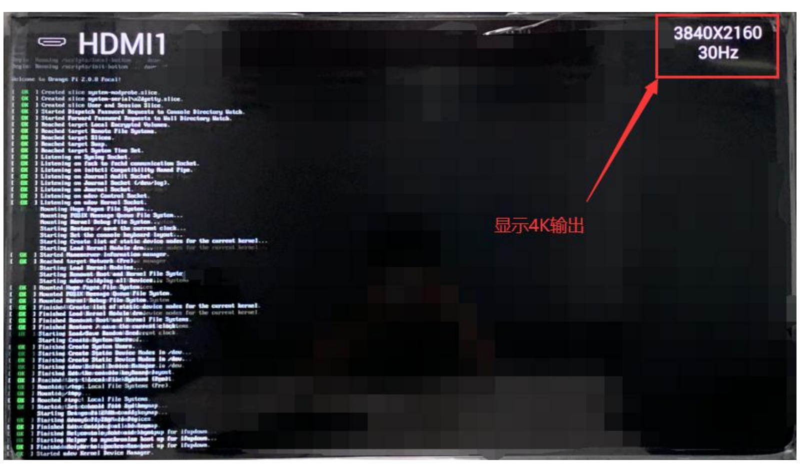
2) disp_mode 变量支持设置的值如下表所示

disp_mode 支持的值	HDMI 分辨率	HDMI 刷新率
480i	720x480	60
576i	720x480	50
480p	720x480	60
576p	720x576	60
720p50	1280x720	50
720p60	1280x720	60
1080i50	1920x1080	50
1080i60	1920x1080	60
1080p24	1920x1080	24
1080p50	1920x1080	50
1080p60	1920x1080	60
2160p24	3840x2160	24
2160p25	3840x2160	25
2160p30	3840x2160	30

3) 将 disp_mode 变量的值修改为想要输出的分辨率，然后重启系统，HDMI 就会输出所设置的分辨率了

4) 如果设置了 HDMI 的输出分辨率为 **2160p24**、**2160p25** 或者 **2160p30**，HDMI 需要接到支持 4K 显示的电视或者显示器才能正常显示

a. 接到 4K 电视显示如下所示



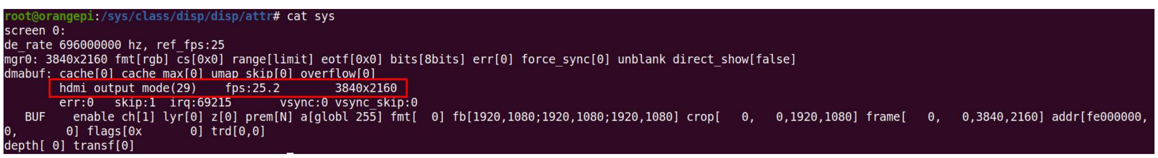
b. 如果接到不支持 4K 显示的电视或者显示器，会无法显示，如下图所示接到

1080p 的电视直接显示**不支持格式**



5) 查看 HDMI 驱动输出分辨率的方法如下所示（下图显示的 HMDI 输出的分辨率为 2160p25），如果显示的分辨率和设置的分辨率一样，说明开发板这端的设置正确

```
root@orangepi:~# cat /sys/class/disp/disp/attr/sys
```



3.12. Linux4.9 系统 Framebuffer 宽度和高度的修改方法

注意：此方法只适用于 linux4.9 内核的系统，linux5.4 内核的系统无法使用

1) 在 linux 系统的/boot/orangepiEnv.txt 中有 fb0_width 和 fb0_height 两个变量，可以通过它们来设置 Framebuffer 的宽度和高度，linux 系统默认设置 fb0_width=1280、fb0_height=720

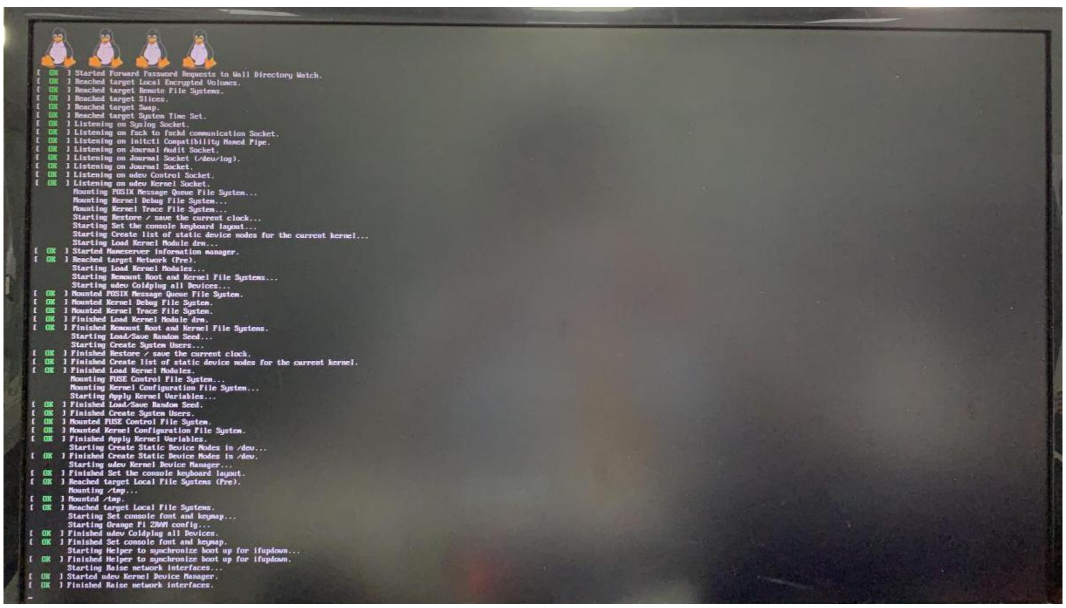
```
root@orangepi:~# cat /boot/orangepiEnv.txt
verbosity=1
console=both
disp_mode=1080p60
fb0_width=1280
fb0_height=720
```

2) fb0_width 和 fb0_height 不同分辨率对应的**参考值**如下所示

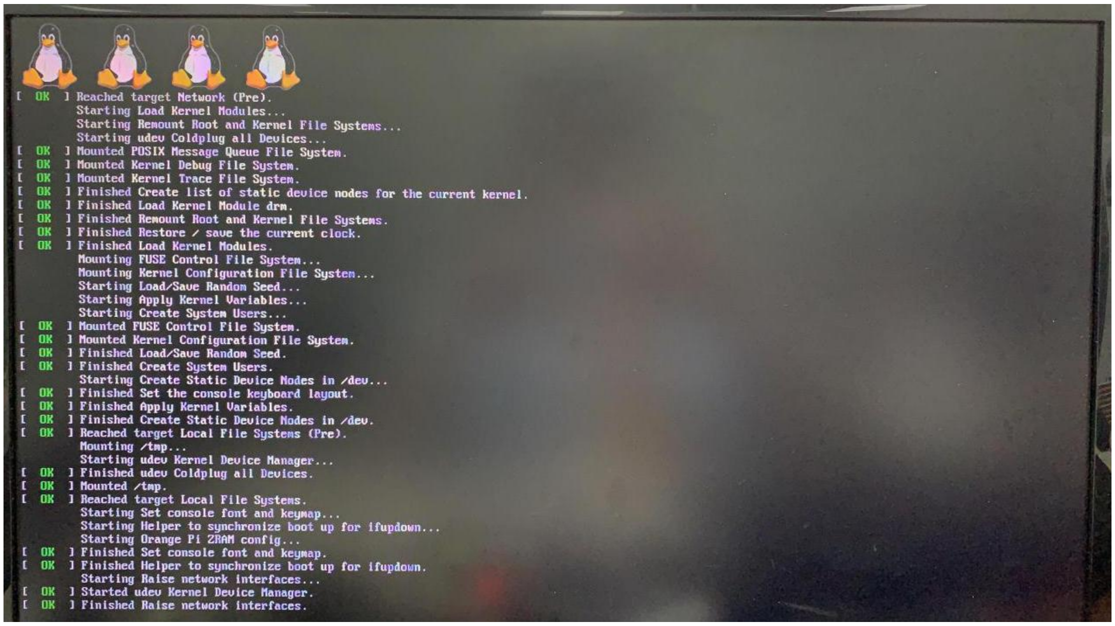
HDMI 分辨率	fb0_width	fb0_height
480p	720	480
576p	720	576
720p	1280	720
1080p	1920	1080
2160p	1920	1080

3) 在相同的 HDMI 分辨下，不同的 fb0_width 和 fb0_height 的显示情况如下所示，当 fb0_width 和 fb0_height 设置的值越大时，屏幕显示的文字就越小，当 fb0_width 和 fb0_height 设置的值越小时，屏幕显示的文字就越大

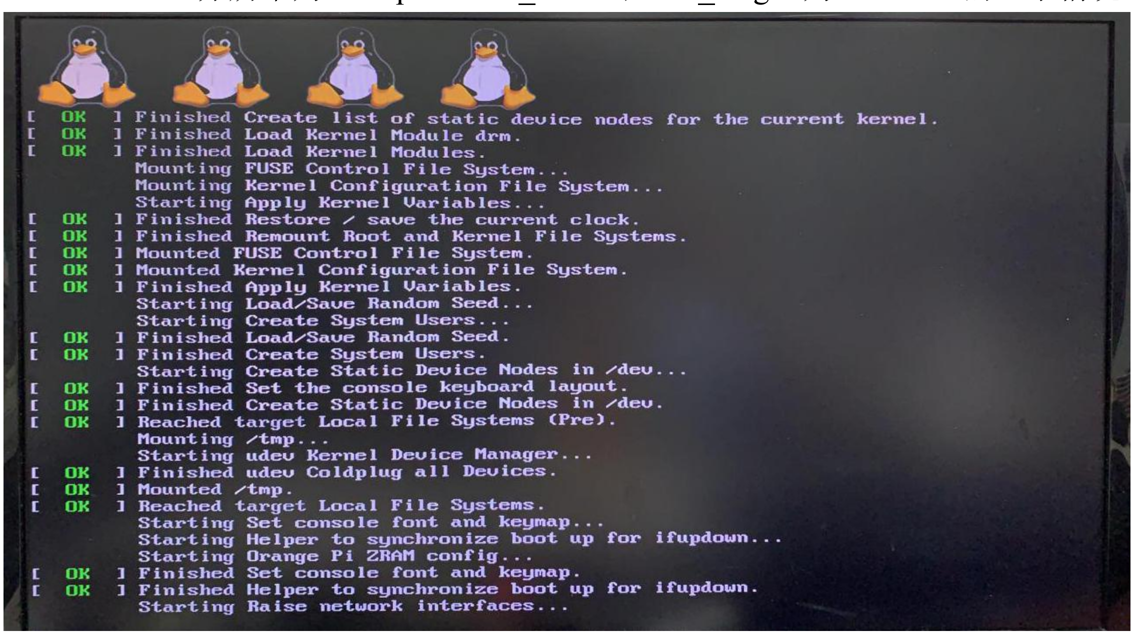
a. HDMI 分辨率为 1080p60，fb0_width 和 fb0_height 为 1920x1080 的显示情况



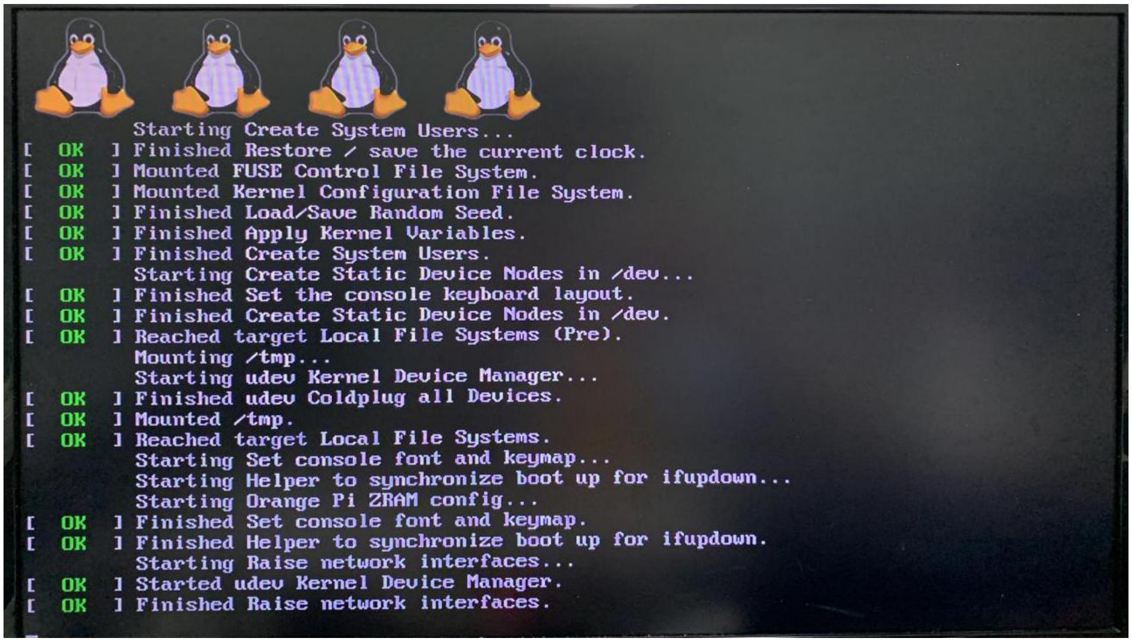
b. HDMI 分辨率为 1080p60，fb0_width 和 fb0_height 为 1280x720 的显示情况



c. HDMI 分辨率为 1080p60，fb0_width 和 fb0_height 为 720x576 的显示情况



d. HDMI 分辨率为 1080p60，fb0_width 和 fb0_height 为 720x480 的显示情况



3. 13. Framebuffer 光标相关的设置

1) Framebuffer 使用的 softcursor，设置光标闪烁或者不闪烁的方法如下所示

```
root@orangepi:~# echo 1 > /sys/class/graphics/fbcon/cursor_blink     #光标闪烁
root@orangepi:~# echo 0 > /sys/class/graphics/fbcon/cursor_blink     #光标不闪烁
```

2) 如果需要隐藏光标，可以在**/boot/orangepiEnv.txt** 的 **extraargs** 变量（**extraargs** 的值会赋值给 **bootargs** 环境变量最终传递给内核）中加入 **vt.global_cursor_default=0**（如果 **vt.global_cursor_default=1** 则是显示光标），然后重启系统就能看到光标已经消失

```
root@orangepi:~# cat /boot/orangepiEnv.txt
verbosity=1
console=both
disp_mode=1080p60
fb0_width=1280
fb0_height=720
extraargs=cma=25M vt.global_cursor_default=0
```

3.14. 香橙派 5 寸 TFT 液晶屏测试

注意：此方法只适用于 linux4.9 内核的系统，linux5.4 内核的系统无法使用

1) 首先准备好香橙派的 5 寸 TFT 液晶屏，屏幕排线和转接板的接线方式如下图所示，请勿接反了，另外请确保屏幕排线和排线插座接触都到位了，如果接触不到位，屏幕输出会有问题



2) 然后使用 HDMI 线将开发板的 HDMI 接口连接到转接板的 HDMI 接口中，屏幕需要单独供电，请在转接板的 Micro USB 接口中插入 5V/2A 的电源



3) 在/boot/orangepiEnv.txt 中修改 HDMI 输出分辨率为 480p，Framebuffer 的长度和宽度可以设置为 800x480，如果设置为 1280x720，TFT 液晶屏显示的字体会很小

```
root@orangepi:~# cat /boot/orangepiEnv.txt
disp_mode=480p
fb0_width=800
fb0_height=480
```

4) 开发板启动后，屏幕就可以看到启动画面了



5) 关闭开发板的电源后，请记得同时关闭屏幕转接板的电源，下次启动开发板时再打开

3. 15. WIFI 连接测试

请不要通过修改/etc/network/interfaces 配置文件的方式来连接 WIFI，通过这种方式连接 WIFI 网络使用会有问题

3. 15. 1. 服务器版镜像通过命令连接 WIFI

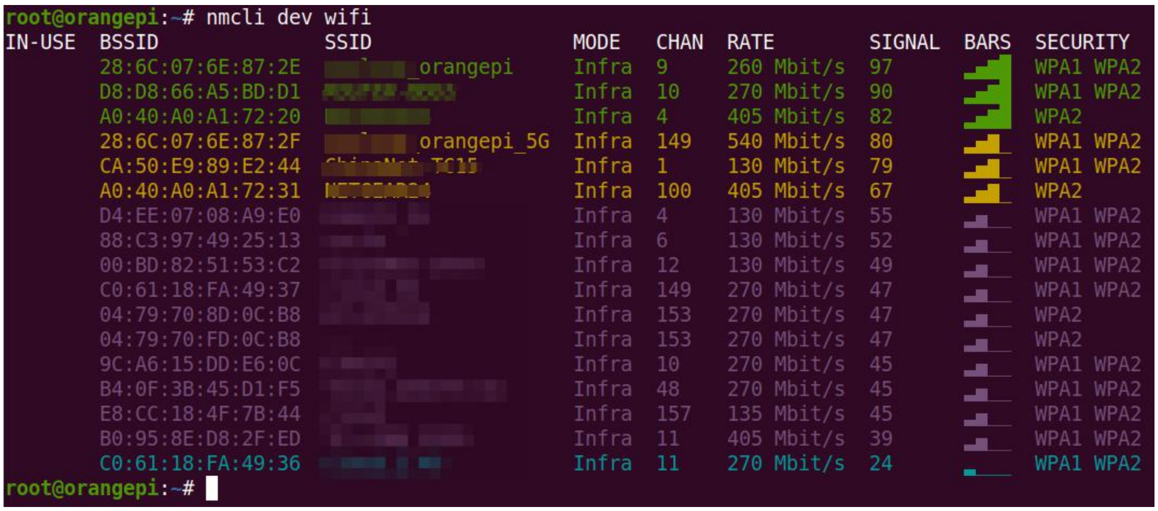
当开发板没有连接以太网，没有连接 HDMI 显示屏，只连接了串口时，推荐使用此小节演示的命令来连接 WIFI 网络。因为 nmtui 在某些串口软件（如 minicom）中只能显示字符，无法正常显示图形界面。当然，如果开发板连接了以太网或者 HDMI 显示屏，也可以使用此小节演示的命令来连接 WIFI 网络的

1) 先登录 linux 系统，有下面三种方式

a. 如果开发板连接了网线，可以通过 ssh 远程登录 linux 系统
a. 如果开发板连接好了调试串口，可以使用串口终端登录 linux 系统
b. 如果连接了开发板到HDMI显示器，可以通过HDMI显示的终端登录到linux系统

2) 首先使用 nmcli dev wifi 命令扫描周围的 WIFI 热点

```
root@orangepi:~# nmcli dev wifi
```



3) 然后使用 **nmcli** 命令连接扫描到的 WIFI 热点，其中：
a. **wifi_name** 需要换成想连接的 WIFI 热点的名字
b. **wifi_passwd** 需要换成想连接的 WIFI 热点的密码

```
root@orangepi:~# nmcli dev wifi connect wifi_name password wifi_passwd
Device 'wlan0' successfully activated with 'cf937f88-ca1e-4411-bb50-61f402eef293'.
```

4) 通过 ifconfig 命令可以查看 wifi 的 IP 地址

```
root@orangepi:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.49  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::76bb:f67d:ef98:2f9a  prefixlen 64  scopeid 0x20<link>
        ether 12:81:3e:a8:58:d8  txqueuelen 1000  (Ethernet)
        RX packets 185  bytes 109447 (109.4 KB)
        RX errors 0  dropped 61  overruns 0  frame 0
        TX packets 27  bytes 14783 (14.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

5) 使用 ping 命令可以测试 wifi 网络的连通性

```
root@orangepi:~# ping www.baidu.com -I wlan0
PING www.a.shifen.com (14.215.177.39) from 192.168.1.119 wlan0: 56(84) bytes of data.
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=3 ttl=56 time=15.0 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=4 ttl=56 time=12.8 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=5 ttl=56 time=13.7 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=6 ttl=56 time=14.9 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=7 ttl=56 time=13.6 ms
^C
```

3.15.2. 服务器版镜像通过图形化方式连接 WIFI

1) 先登录 linux 系统，有下面三种方式

a. 如果开发板连接了网线，可以通过 ssh 远程登录 linux 系统

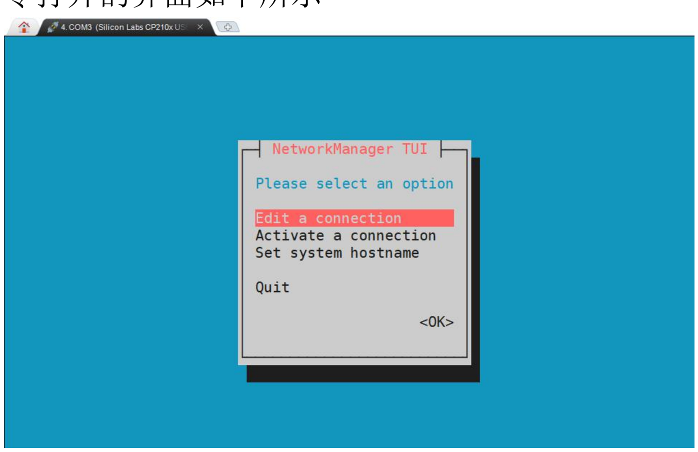
b. 如果开发板连接好了调试串口，可以使用串口终端登录 linux 系统（串口软件请使用 MobaXterm，使用 minicom 无法显示图形界面）

c. 如果开发板连接了开发板到 HDMI 显示器，可以通过 HDMI 显示的终端登录到 linux 系统

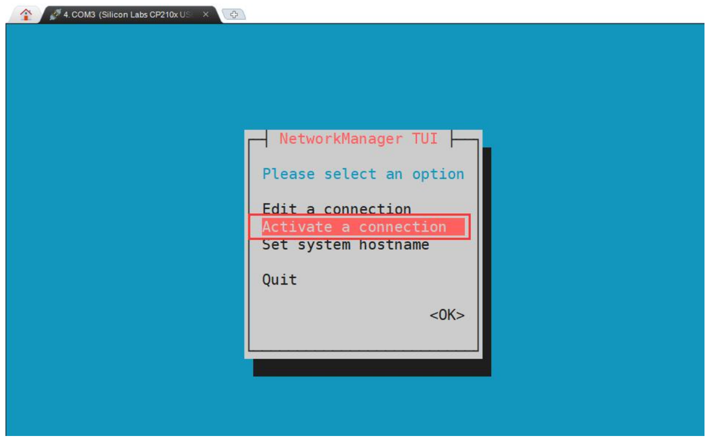
2) 然后在命令行中输入 nmtui 命令打开 wifi 连接的界面

```
root@orangepi:~# nmtui
```

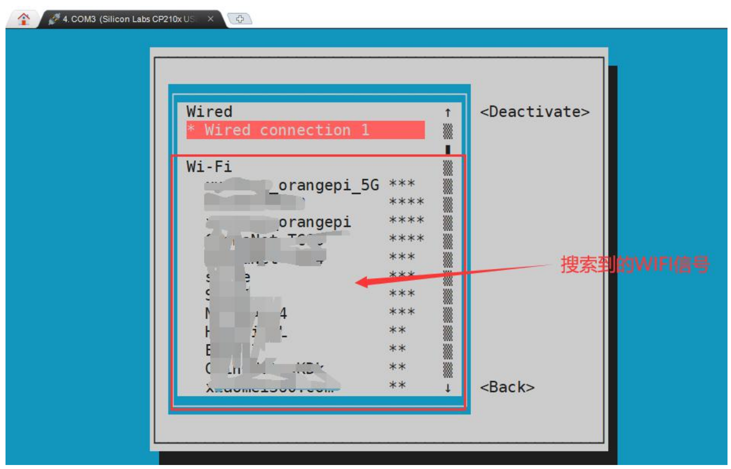
3) 输入 nmtui 命令打开的界面如下所示



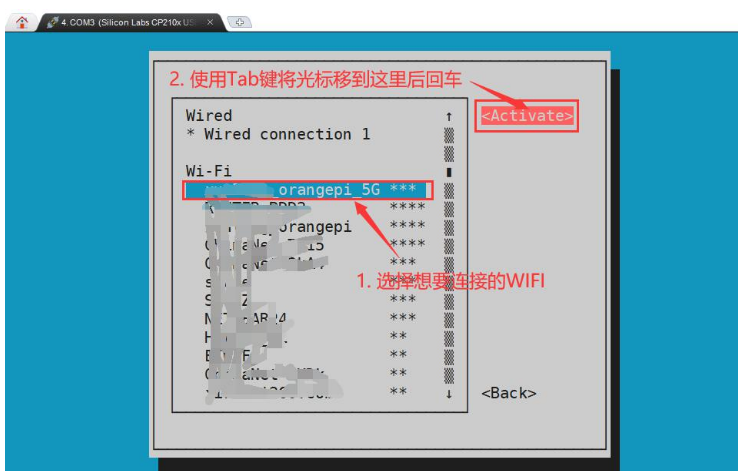
4) 选择 **Activate a connect** 后回车



5) 然后就能看到所有搜索到的 WIFI 热点

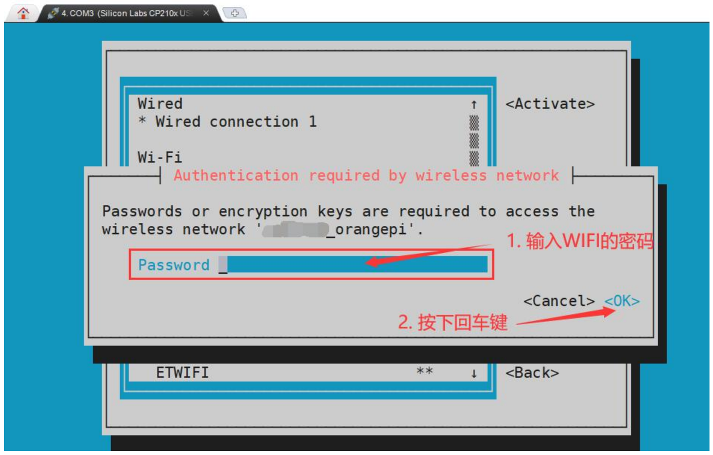


6) 选择想要连接的 WIFI 热点后再使用 Tab 键将光标定位到 **Activate** 后回车



7) 然后会弹出输入密码的对话框，在 **Pssword** 内输入对应的密码然后回车就会开始

连接 WIFI



8) WIFI 连接成功后会在已连接的 WIFI 名称前显示一个“*”



9) 通过 ifconfig 命令可以查看 wifi 的 IP 地址

```
root@orangepi:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.49  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::76bb:f67d:ef98:2f9a  prefixlen 64  scopeid 0x20<link>
        ether 12:81:3e:a8:58:d8  txqueuelen 1000  (Ethernet)
        RX packets 185  bytes 109447 (109.4 KB)
        RX errors 0  dropped 61  overruns 0  frame 0
        TX packets 27  bytes 14783 (14.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

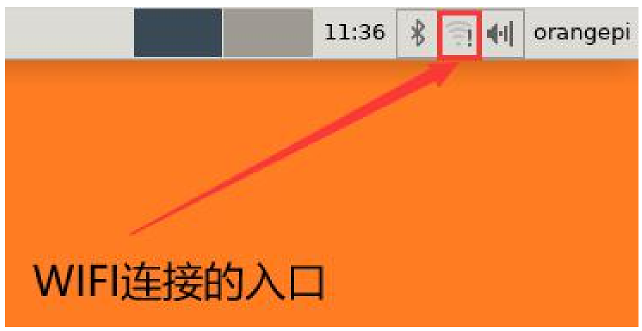
10) 使用 ping 命令可以测试 wifi 网络的连通性

```
root@orangepi:~# ping www.baidu.com -I wlan0
```

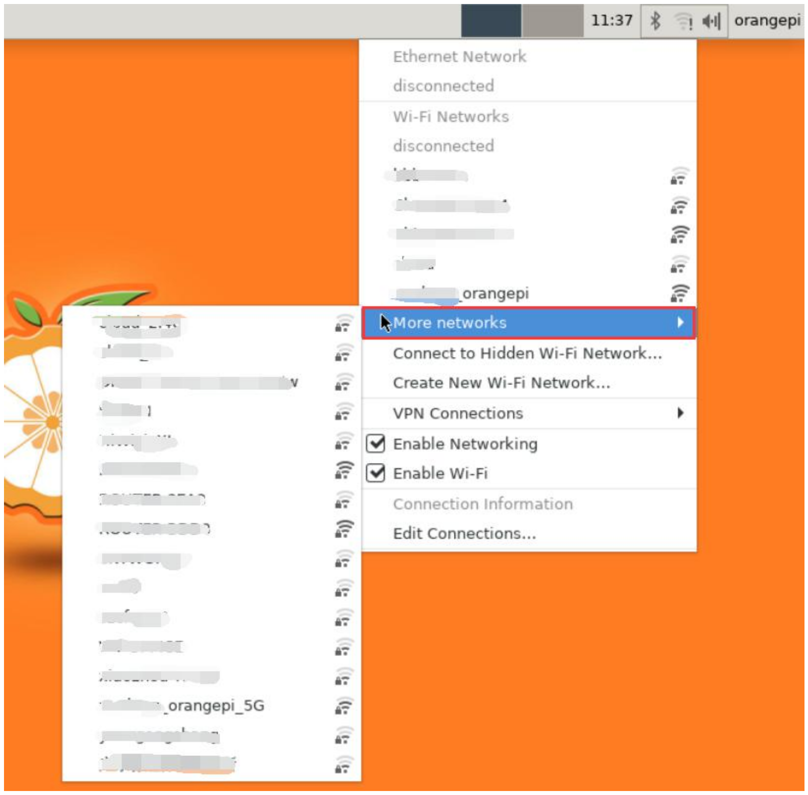
```
PING www.a.shifen.com (14.215.177.39) from 192.168.1.119 wlan0: 56(84) bytes of data.
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=3 ttl=56 time=15.0 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=4 ttl=56 time=12.8 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=5 ttl=56 time=13.7 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=6 ttl=56 time=14.9 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=7 ttl=56 time=13.6 ms
^C
```

3.15.3. 桌面版镜像的测试方法

1) 点击桌面右上角的网络配置图标（测试 WIFI 时请不要连接网线）



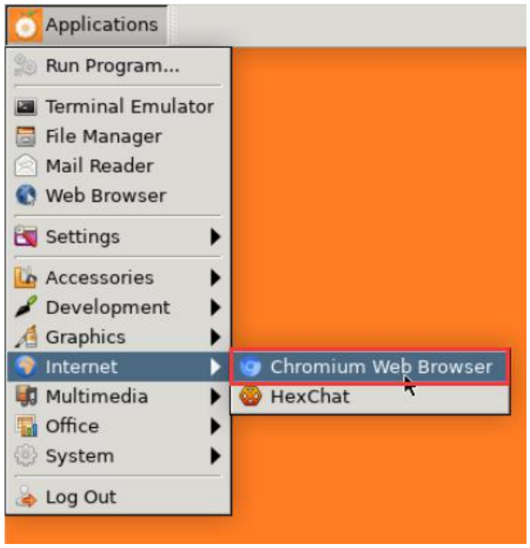
2) 在弹出的下拉框中点击 **More networks** 可以看到所有扫描到的 WIFI 热点，然后选择想要连接的 WIFI 热点



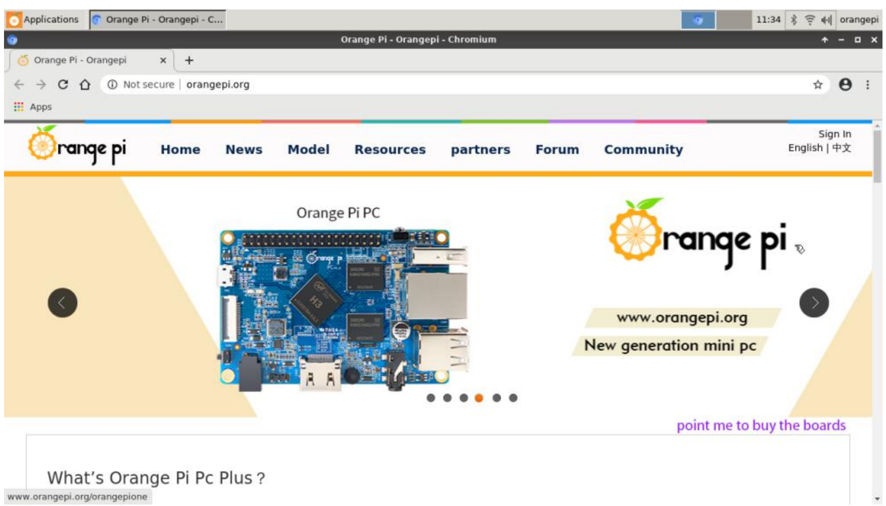
3) 然后输入 WIFI 热点的密码，再点击 **Connect** 就会开始连接 WIFI



4) 连接好 WIFI 后，可以打开浏览器查看是否能上网，浏览器的入口如下图所示



5) 打开浏览器后如果能看到 Orange Pi 网站的页面 ，或者能打开其他网页说明 WIFI 连接正常



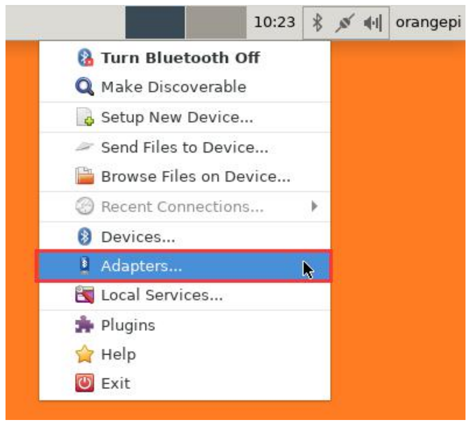
3.16. 蓝牙使用方法

3.16.1. 桌面版镜像的测试方法

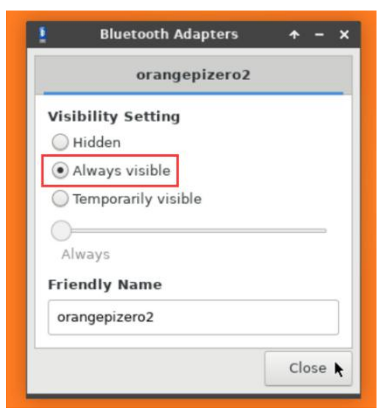
1) 点击桌面右上角的蓝牙图标



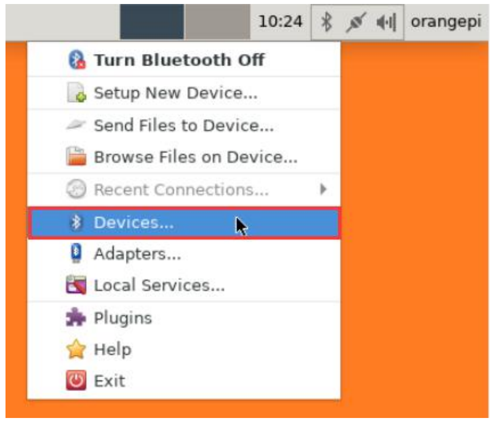
2) 然后选择适配器



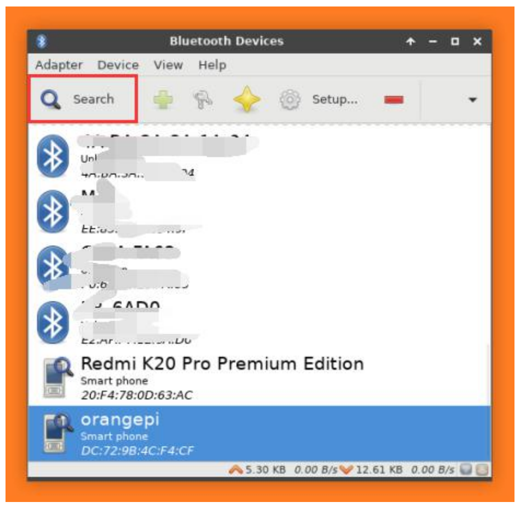
3) 在蓝牙的适配器设置界面中设置 **Visibility Setting** 为 **Always visible**，然后点击 **close** 关闭



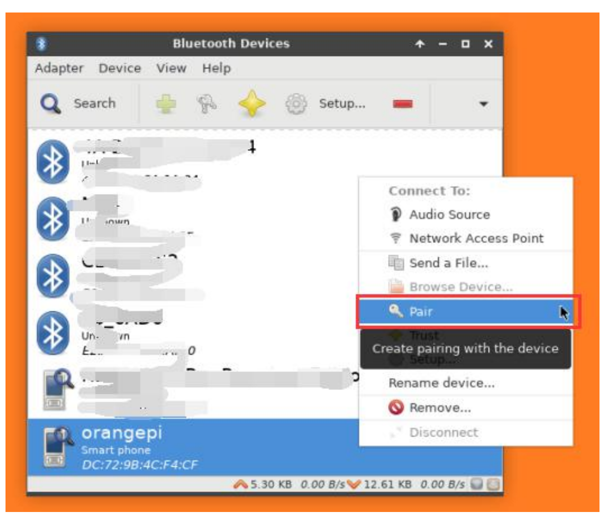
4) 然后打开蓝牙设备的配置界面



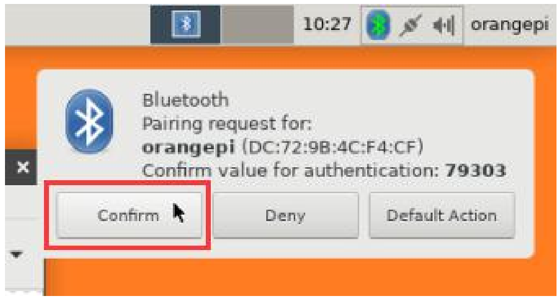
5) 点击 **Search** 即可开始扫描周围的蓝牙设备



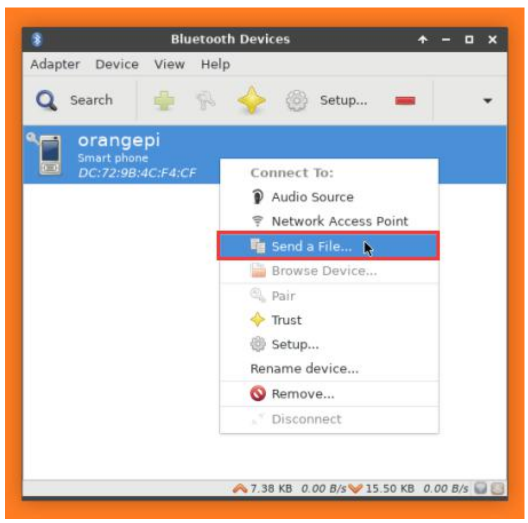
6) 然后选择想要连接的蓝牙设备，再点击鼠标右键就会弹出对此蓝牙设备的操作界面，选择 **Pair** 即可开始配对，这里演示的是和 Android 手机配对



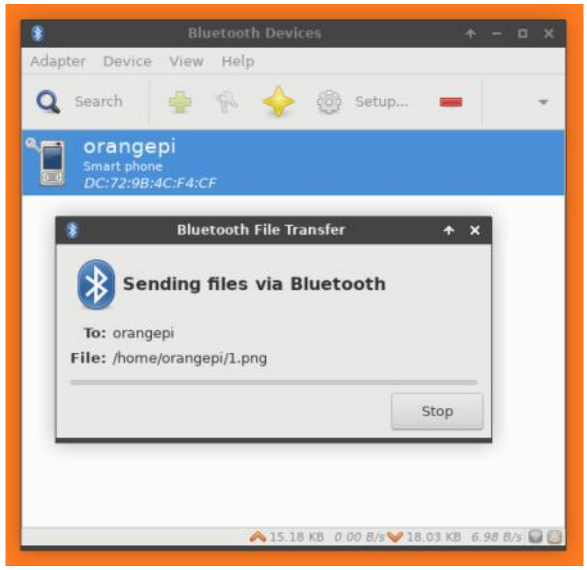
7) 配对时，桌面的右上角会弹出配对确认框，选择 **Confirm** 确认即可，此时手机上也同样需要进行确认



8) 和手机配对完后，可以选择已配对的蓝牙设备，然后右键选择 **Send a File** 即可开始给手机发送一张图片



9) 发送图片的界面如下所示



3.16.2. 服务器版镜像的使用方法

1) linux 系统启动时会运行 ap6256-bluetooth.service 服务来初始化蓝牙设备，进入系统后可以通过 hciconfig 命令来查看是否存在蓝牙的设备节点，如果存在，说明蓝牙初始化正常

```
root@orangepi:~# hciconfig -a
hci0:    Type: Primary    Bus: UART
         BD Address: 10:11:12:13:14:15    ACL MTU: 1021:8    SCO MTU: 240:3
         UP RUNNING
         RX bytes:646 acl:0 sco:0 events:37 errors:0
         TX bytes:2650 acl:0 sco:0 commands:37 errors:0
         Features: 0xbf 0xff 0x8d 0xfe 0xdb 0x3d 0x7b 0xc7
         Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
         Link policy:
         Link mode: SLAVE ACCEPT
         Name: 'orangepi3'
         Class: 0x000000
         Service Classes: Unspecified
         Device Class: Miscellaneous,
         HCI Version: 5.0 (0x9)    Revision: 0x400
         LMP Version: 5.0 (0x9)    Subversion: 0x400
         Manufacturer: Spreadtrum Communications Shanghai Ltd (492)
```

2) 使用 bluetoothctl 扫描蓝牙设备

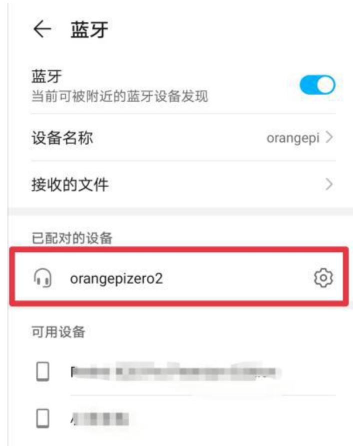
```
root@orangepi:~# bluetoothctl
[NEW] Controller 10:11:12:13:14:15 orangepi3 [default]
Agent registered
[bluetooth]# power on            #使能控制器
Changing power on succeeded
[bluetooth]# discoverable on      #设置控制器为可被发现的
Changing discoverable on succeeded
[CHG] Controller 10:11:12:13:14:15 Discoverable: yes
[bluetooth]# pairable on     #设置控制器为可配对的
Changing pairable on succeeded
[bluetooth]# scan on           #开始扫描周围的蓝牙设备
Discovery started
[CHG] Controller 10:11:12:13:14:15 Discovering: yes
[NEW] Device 76:60:79:29:B9:31 76-60-79-29-B9-31
[NEW] Device 9C:2E:A1:42:71:11  小米手机
[NEW] Device DC:72:9B:4C:F4:CF orangepi
[bluetooth]# scan off          #扫描到想连接的蓝牙设备后就可以关闭扫描了，然后记下蓝牙设备的 MAC 地址，这里测试的蓝牙设备为 Android 手机，蓝牙的名字为 orangepi，对应的 MAC 地址为 DC:72:9B:4C:F4:CF
Discovery stopped
[CHG] Controller 10:11:12:13:14:15 Discovering: no
[CHG] Device DC:72:9B:4C:F4:CF RSSI is nil
```

3) 扫描到想配对的设备后就可以进行配对了，配对需要使用设备的 MAC 地址

```
[bluetooth]# pair DC:72:9B:4C:F4:CF       #使用扫描到的蓝牙设备的 MAC 地址进行配对
Attempting to pair with DC:72:9B:4C:F4:CF
[CHG] Device DC:72:9B:4C:F4:CF Connected: yes
Request confirmation
[leeb1m[agent] Confirm passkey 764475 (yes/no): yes    #在这里输入 yes，在手机上也需要确认
[CHG] Device DC:72:9B:4C:F4:CF Modalias: bluetooth:v010Fp107Ed1436
[CHG] Device DC:72:9B:4C:F4:CF UUIDs: 0000046a-0000-1000-8000-00805f9b34fb
[CHG] Device DC:72:9B:4C:F4:CF ServicesResolved: yes
[CHG] Device DC:72:9B:4C:F4:CF Paired: yes
```

```
Pairing successful      #提示配对成功
[CHG] Device DC:72:9B:4C:F4:CF ServicesResolved: no
[CHG] Device DC:72:9B:4C:F4:CF Connected: no
```

4) 配对成功后，手机蓝牙界面的显示如下所示



5) 连接蓝牙设备需要安装 **pulseaudio-module-bluetooth** 软件包，然后再启动 **pulseaudio** 服务

```
root@orangepi:~# apt update
root@orangepi:~# apt -y install pulseaudio-module-bluetooth
root@orangepi:~# pulseaudio --start
```

6) 连接蓝牙设备的方法

```
root@orangepi:~# bluetoothctl
Agent registered
[bluetooth]# paired-devices       #查看已配对的蓝牙设备的 MAC 地址
Device DC:72:9B:4C:F4:CF orangepi
[bluetooth]# connect DC:72:9B:4C:F4:CF     #使用 MAC 地址连接蓝牙设备
Attempting to connect to DC:72:9B:4C:F4:CF
[CHG] Device DC:72:9B:4C:F4:CF Connected: yes
Connection successful
[CHG] Device DC:72:9B:4C:F4:CF ServicesResolved: yes
[CHG] Controller 10:11:12:13:14:15 Discoverable: no
[orangepi]#                #出现这个提示符说明连接成功
```

7) 连接完蓝牙设备后，Android 手机的蓝牙配置界面就可以看到**已连接用于通话和媒体的音频**的提示



3.17. 板载 LED 灯测试说明

1) 开发板上有两个 LED 灯，一个黄灯，一个红灯，系统启动时 LED 灯默认显示情况如下所示

	黄灯	红灯
u-boot 启动阶段	灭	亮
内核启动到进入系统	亮	灭
GPIO 口	PL7	PL4

2) 设置黄灯亮灭和闪烁的方法如下所示

a. 首先进入黄灯的设置目录

```
root@orangepi:~# cd /sys/class/leds/orangepi\:yellow\:status
```

b. 设置黄灯熄灭的命令如下

```
root@orangepi:/sys/class/leds/orangepi:yellow:status# echo 0 > brightness
```

c. 设置黄灯常亮的命令如下

```
root@orangepi:/sys/class/leds/orangepi:yellow:status# echo 1 > brightness
```

d. 设置黄灯闪烁的命令如下

```
root@orangepi:/sys/class/leds/orangepi:yellow:status# echo heartbeat > trigger
```

e. 设置黄灯停止闪烁的命令如下

```
root@orangepi:/sys/class/leds/orangepi:yellow:status# echo none > trigger
```

3) 设置红灯亮灭和闪烁的方法如下所示

a. 首先进入红灯的设置目录

```
root@orangepi:~# cd /sys/class/leds/orangepi\:red\:status
```

b. 设置红灯熄灭的命令如下

```
root@orangepi:/sys/class/leds/orangepi:red:status# echo 0 > brightness
```

c. 设置红灯常亮的命令如下

```
root@orangepi:/sys/class/leds/orangepi:red:status# echo 1 > brightness
```

d. 设置红灯闪烁的命令如下

```
root@orangepi:/sys/class/leds/orangepi:red:status# echo heartbeat > trigger
```

e. 设置红灯停止闪烁的命令如下

```
root@orangepi:/sys/class/leds/orangepi:red:status# echo none > trigger
```

3.18. USB 接口测试

3.18.1. 连接鼠标或键盘测试

1) 将 USB 接口的键盘插入 Orange Pi 开发板的 USB 接口中

2) 连接 Orange Pi 开发板到 HDMI 显示器

3) 如果鼠标或键盘能正常操作系统说明 USB 接口使用正常（鼠标只有在桌面版的系统中才能使用）

3.18.2. 连接 USB 存储设备测试

1) 首先将 U 盘插入 Orange Pi 开发板的 USB 接口中

2) 执行下面的命令如果能看到 sdX 的输出说明 U 盘识别成功

```
root@orangepi:~# cat /proc/partitions | grep "sd*"
major minor    #blocks    name
   8          0      30044160 sda
   8          1      30043119 sda1
```

3) 使用 mount 命令可以将 U 盘挂载到/mnt 中，然后就能查看 U 盘中的文件了

```
root@orangepi:~# mount /dev/sda1 /mnt/
root@orangepi:~# ls /mnt/
```

```
test.txt
```

4) 挂载完后通过 df -h 命令就能查看 U 盘的容量使用情况和挂载点

```
root@orangepi:~# df -h | grep "sd"
/dev/sda1          29G   208K    29G     1% /mnt
```

3.19. USB 无线网卡测试

注意：此方法只适用于 linux5.4 内核的系统，linux4.9 内核的系统没有调试

linux5.4 系统目前**测试过的**能用的 USB 无线网卡如下所示，其他型号的 USB 无线网卡请自行测试，如果无法使用就需要移植对应的 USB 无线网卡驱动

序号	型号
1	RTL8723BU
2	RTL8821CU

3.19.1.1. RTL8723BU 测试

1) 首先将 RTL8723BU 无线网卡模块插入开发板的 USB 接口中

2) 然后 linux 系统会自动加载 RTL8723BU 相关的内核模块，通过 lsmod 命令可以看到下面输出

```
root@orangepi:~# lsmod | grep "rtl8"
rtl8xxxu               118784  0
mac80211               503808  1 rtl8xxxu
```

3) 通过 dmesg 命令可以看到 RTL8723BU 模块的加载信息

```
root@orangepi:~# dmesg | tail
[ 1583.908354] usb 3-1.1: 1e0: ff ff ff ff ff ff ff ff
[ 1583.908357] usb 3-1.1: 1e8: ff ff ff ff ff ff ff ff
[ 1583.908360] usb 3-1.1: 1f0: ff ff ff ff ff ff ff ff
[ 1583.908363] usb 3-1.1: 1f8: ff ff ff ff ff ff ff ff
[ 1583.908369] usb 3-1.1: RTL8723BU rev E (SMIC) 1T1R, TX queues 3, WiFi=1,
BT=1, GPS=0, HI PA=0
```

```
[ 1583.908373] usb 3-1.1: RTL8723BU MAC: 00:13:ef:f4:58:ae
[ 1583.908377] usb 3-1.1: rtl8xxxu: Loading firmware rtlwifi/rtl8723bu_nic.bin
[ 1583.908536] usb 3-1.1: Firmware revision 35.0 (signature 0x5301)
[ 1584.013655] Bluetooth: hci1: RTL: fw version 0x1e4cc3ff
[ 1584.813524] rtl8xxxu 3-1.1:1.2 wlx0013eff458ae: renamed from wlan1
```

4) 然后通过 ifconfig 命令可以看到 RTL8723BU WIFI 的设备节点，WIFI 的连接和测试方法请参看 WIFI 连接测试一节，这里不再赘述

```
root@orangepi:~# ifconfig wlx0013eff458ae
wlx0013eff458ae: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 00:13:ef:f4:58:ae  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

5) 然后通过 hciconfig 命令可以看到两个蓝牙设备，其中 Bus 类型为 USB 的节点就是 RTL8723BU 的蓝牙节点，蓝牙的测试方法请参看蓝牙使用方法一节，这里不再赘述

```
root@orangepi:~# hciconfig
hci1:   Type: Primary  Bus: USB
        BD Address: 00:13:EF:F4:58:AE  ACL MTU: 1021:8  SCO MTU: 255:16
        UP RUNNING PSCAN ISCAN
        RX bytes:3994 acl:0 sco:0 events:206 errors:0
        TX bytes:29459 acl:0 sco:0 commands:173 errors:0

hci0:   Type: Primary  Bus: UART
        BD Address: 10:11:12:13:14:15  ACL MTU: 1021:8  SCO MTU: 240:3
        UP RUNNING
        RX bytes:2981 acl:0 sco:0 events:127 errors:0
        TX bytes:5423 acl:0 sco:0 commands:61 errors:0
```

3.19.1.2. RTL8821CU 测试

1) 首先将 RTL8821CU 无线网卡模块插入开发板的 USB 接口中

2) 然后 linux 系统会自动加载 RTL8821CU 相关的内核模块，通过 lsmod 命令可以看到下面输出

```
root@orangepi:~# lsmod | grep "8821"
8821cu                  1941504   0
cfg80211                 356352   3 8821cu,brcmfmac,mac80211
```

3) 通过 dmesg 命令可以看到 RTL8821CU 模块的加载信息

```
root@orangepi:~# dmesg | tail
[ 1835.290228] usb 3-1.1: Product: 802.11ac NIC
[ 1835.290239] usb 3-1.1: Manufacturer: Realtek
[ 1835.290249] usb 3-1.1: SerialNumber: 123456
[ 1835.371795] Bluetooth: hci1: RTL: examining hci_ver=08 hci_rev=000c lmp_ver=08
lmp_subver=8821
[ 1835.372753] Bluetooth: hci1: RTL: rom_version status=0 version=1
[ 1835.372771] Bluetooth: hci1: RTL: loading rtl_bt/rtl8821c_fw.bin
[ 1835.373136] Bluetooth: hci1: RTL: loading rtl_bt/rtl8821c_config.bin
[ 1835.373349] Bluetooth: hci1: RTL: cfg_sz 10, total sz 21678
[ 1835.784650] Bluetooth: hci1: RTL: fw version 0x826ca99e
[ 1835.816409] rtl8821cu 3-1.1:1.2 wlxd0c0bf8742cd: renamed from wlan1
```

4) 然后通过 ifconfig 命令可以看到 RTL8821CU WIFI 的设备节点，WIFI 的连接和测试方法请参看 WIFI 连接测试一节，这里不再赘述

```
root@orangepi:~# ifconfig wlxd0c0bf8742cd
wlxd0c0bf8742cd: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether d0:c0:bf:87:42:cd  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

5) 然后通过 hciconfig 命令可以看到两个蓝牙设备，其中 Bus 类型为 USB 的节点就

是 RTL8821CU 的蓝牙节点，蓝牙的测试方法请参看蓝牙使用方法一节，这里不再赘述

```
root@orangepi:~# hciconfig
hci1:    Type: Primary   Bus: USB
         BD Address: D0:C0:BF:87:42:CE   ACL MTU: 1021:8   SCO MTU: 255:12
         UP RUNNING
         RX bytes:1343 acl:0 sco:0 events:137 errors:0
         TX bytes:24227 acl:0 sco:0 commands:137 errors:0

hci0:    Type: Primary   Bus: UART
         BD Address: 43:45:C5:00:1F:AC   ACL MTU: 1021:8   SCO MTU: 64:1
         UP RUNNING
         RX bytes:1941 acl:0 sco:0 events:66 errors:0
         TX bytes:2816 acl:0 sco:0 commands:66 errors:0
```

3. 20. USB 网卡测试

1) 目前**测试过**能用的 USB 网卡如下所示

序号	型号
1	RTL8152B USB 百兆网卡
2	RTL8153 USB 千兆网卡

2) 首先将 USB 网卡插入开发板的 USB 接口中，然后在 USB 网卡中插入网线，确保网线能正常上网，如果通过 dmesg 命令可以看到下面的 log 信息，说明 USB 网卡识别正常

```
root@orangepi:~# dmesg | tail
[  121.985016] usb 3-1: USB disconnect, device number 2
[  126.873772] sunxi-ehci 5311000.ehci3-controller: ehci_irq: highspeed device connect
[  127.094054] usb 3-1: new high-speed USB device number 3 using sunxi-ehci
[  127.357472] usb 3-1: reset high-speed USB device number 3 using sunxi-ehci
[  127.557960] r8152 3-1:1.0 eth1: v1.08.9
[  127.602642] r8152 3-1:1.0 enx00e04c362017: renamed from eth1
[  127.731874] IPv6: ADDRCONF(NETDEV_UP): enx00e04c362017: link is not ready
[  127.763031] IPv6: ADDRCONF(NETDEV_UP): enx00e04c362017: link is not ready
[  129.892465] r8152 3-1:1.0 enx00e04c362017: carrier on
```

```
[    129.892583] IPv6: ADDRCONF(NETDEV_CHANGE): enx00e04c362017: link becomes ready
```

3) 然后通过 ifconfig 命令可以看到 USB 网卡的设备节点，以及自动分配的 IP 地址

```
root@orangepi:~# ifconfig
enx00e04c362017: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>    mtu 1500
        inet 192.168.1.177   netmask 255.255.255.0   broadcast 192.168.1.255
        inet6 fe80::681f:d293:4bc5:e9fd   prefixlen 64   scopeid 0x20<link>
        ether 00:e0:4c:36:20:17   txqueuelen 1000   (Ethernet)
        RX packets 1849   bytes 134590 (134.5 KB)
        RX errors 0   dropped 125   overruns 0   frame 0
        TX packets 33   bytes 2834 (2.8 KB)
        TX errors 0   dropped 0 overruns 0   carrier 0   collisions 0
```

4) 测试网络连通性的命令如下

```
root@orangepi:~# ping www.baidu.com -I enx00e04c362017
PING www.a.shifen.com (14.215.177.38) from 192.168.1.12 eth0: 56(84) bytes of data.
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=1 ttl=56 time=6.74 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=2 ttl=56 time=6.80 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=3 ttl=56 time=6.26 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=4 ttl=56 time=7.27 ms
^C
--- www.a.shifen.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 6.260/6.770/7.275/0.373 ms
```

3.21. USB 摄像头测试

1) 首先将 USB 摄像头插入到 Orange Pi 开发板的 USB 接口中

2) 然后通过 lsmod 命令可以看到内核自动加载了下面的模块

a. linux4.9 系统显示如下所示

```
root@orangepi:~# lsmod | grep "uvc"
Module                  Size   Used by
```

```
uvcvideo                 106496  0
```

b. linux5.4 系统显示如下所示

```
root@orangepi:~# lsmod | grep "uvc"
uvcvideo                 102400  0
videobuf2_vmalloc         20480  1 uvcvideo
videobuf2_v4l2            28672  1 uvcvideo
videobuf2_common          53248  2 videobuf2_v4l2,uvcvideo
videodev                 233472  3 videobuf2_v4l2,uvcvideo,videobuf2_common
mc                        49152  4
videodev,videobuf2_v4l2,uvcvideo,videobuf2_common
```

3) 通过 v4l2-ctl（**注意 v4l2 中的 l 是小写字母 l，不是数字 1**）命令可以看到 USB 摄像头的设备节点信息为/dev/video0

```
root@orangepi:~# apt update
root@orangepi:~# apt install v4l-utils
root@orangepi:~# v4l2-ctl   --list-devices
USB 2.0 Camera (usb-xhci-hcd.0.auto-1.2):
        /dev/video0
```

4) 使用 fswebcam 测试 USB 摄像头

a. 安装 fswebcam

```
root@orangepi:~# apt update
root@orangepi:~# apt-get install fswebcam
```

b. 安装完 fswebcam 后可以使用下面的命令来拍照

a) -d 选项用于指定 USB 摄像头的设备节点

b) --no-banner 用于去除照片的水印

c) -r 选项用于指定照片的分辨率

d) -S 选项用设置于跳过前面的帧数

e) ./image.jpg 用于设置生成的照片的名字和路径

```
root@orangepi:~# fswebcam -d /dev/video0 --no-banner -r 1280x720 -S 5 ./image.jpg
```

c. 在服务器版的 linux 系统中，拍完照后可以使用 scp 命令将拍好的图片传到 Ubuntu PC 上镜像观看

```
root@orangepi:~# scp image.jpg test@192.168.1.55:/home/test  （根据实际情况修改IP 地址和路径）
```

d. 在桌面版的 linux 系统中，可以通过 HDMI 显示器直接查看拍摄的图片

5) 使用 motion 测试 USB 摄像头

a. 安装摄像头测试软件 motion

```
root@orangepi:~# apt update
root@orangepi:~# apt install motion
```

b. 修改 **/etc/default/motion** 的配置，将 start_motion_daemon=no 修改为 start_motion_daemon=yes

```
root@orangepi:~# sed -i "s/start_motion_daemon=no/start_motion_daemon=yes/" \
/etc/default/motion        (这是一条命令)
```

c. 修改**/etc/motion/motion.conf** 的配置

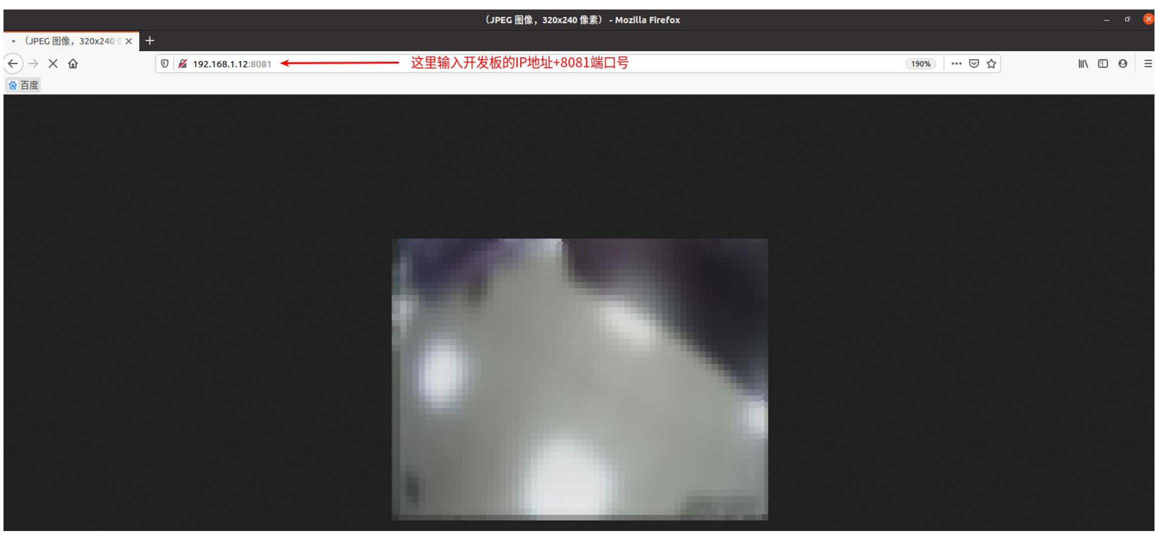
```
root@orangepi:~# sed -i "s/stream_localhost on/stream_localhost off/" \
/etc/motion/motion.conf        (这是一条命令)
```

d. 然后重启 motion 服务

```
root@orangepi:~# /etc/init.d/motion restart
[ ok ] Restarting motion (via systemctl): motion.service.
```

e. 使用 motion 前请先确保 Orange Pi 开发板能正常连接网络，然后通过 ifconfig 命令获取开发板的 IP 地址

f. 然后在和开发板同一局域网的 Ubuntu PC 或者 Windows PC 或者手机的火狐浏览器中输入【**开发板的 IP 地址:8081**】就能看到摄像头输出的视频了



6) 使用 mjpg-streamer 测试 USB 摄像头

a. 下载 mjpg-streamer

```
root@orangepi:~# git clone https://github.com/jacksonliam/mjpg-streamer
```

b. 安装依赖的软件包

```
root@orangepi:~# apt-get install cmake libjpeg8-dev
```

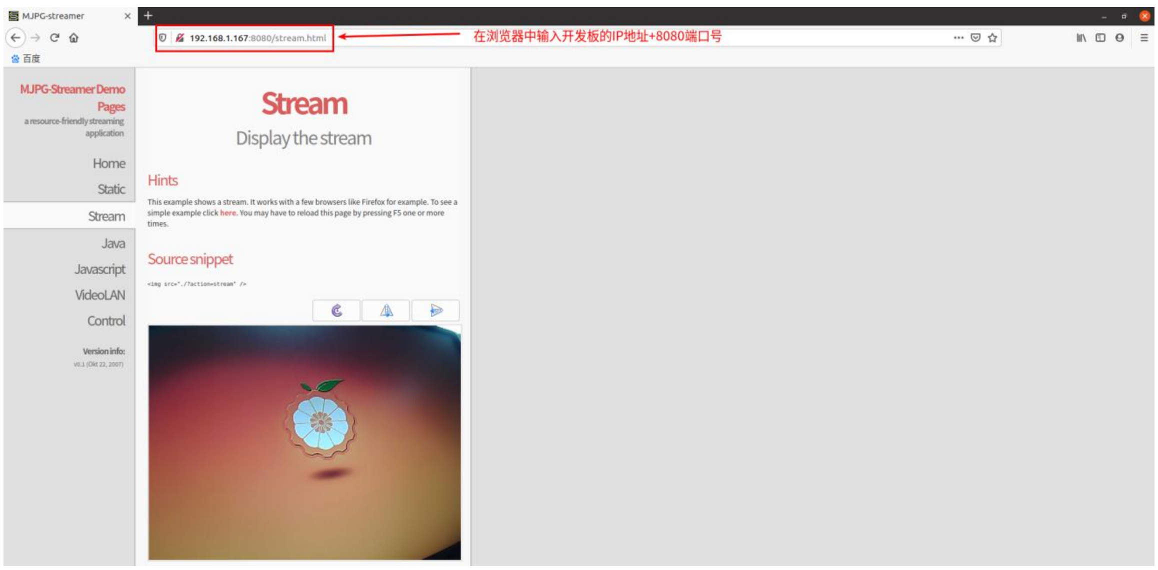
c. 编译安装 mjpg-streamer

```
root@orangepi:~# cd mjpg-streamer/mjpg-streamer-experimental
root@orangepi:~/mjpg-streamer/mjpg-streamer-experimental# make
root@orangepi:~/mjpg-streamer/mjpg-streamer-experimental# make install
```

d. 然后输入下面的命令启动 mjpg_streamer

```
root@orangepi:~/mjpg-streamer/mjpg-streamer-experimental# export
LD_LIBRARY_PATH=.          （这是一条命令）
root@orangepi:~/mjpg-streamer/mjpg-streamer-experimental# ./mjpg_streamer -i
"./input_uvc.so -d /dev/video0 -u -f 30" -o "./output_http.so -w ./www"
```

e. 然后在和开发板同一局域网的 Ubuntu PC 或者 Windows PC 或者手机的浏览器中输入**【开发板的 IP 地址:8080】**就能看到摄像头输出的视频了

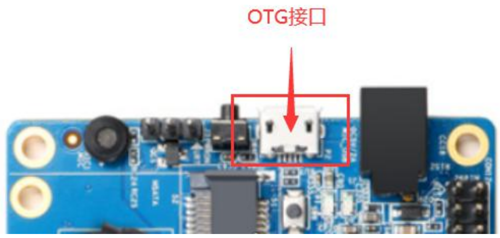


f. 推荐使用 mjpg-streamer 来测试 USB 摄像头，比 motion 流畅很多，使用 mjpg-streamer 感觉不到任何卡顿

3.22. linux4.9 系统 OTG 功能测试

3.22.1. OTG 功能说明

1) OTG 接口的位置如下图所示，此接口既可设置为 USB Host 模式来接 U 盘、鼠标、键盘、USB 摄像头等设备，也可设置为 USB device 模式将开发板连接到电脑来通信



2) 当 OTG 接口设置 USB Host 模式时，需要准备一根 micro USB 转 otg 的转接线才能接 USB 设备



3) 使用 OTG 接口时，请使用 DC 电源来给开发板供电，电脑的 USB 接口无法提供足够的功率来驱动开发板



3.22.2. OTG 接口设置为 USB Host 模式的使用方法

1) linux4.9 镜像 OTG 接口默认设置为 usb_host 模式，可以通过下面的命令来查看 **otg_role** 的状态

```
root@orangepi:~# cat /sys/devices/platform/soc/usbc0/otg_role
usb_host
```

2) 如果 otg_role 没有设置为 usb_host 模式，可以使用下面的命令打开

```
root@orangepi:~# cat /sys/devices/platform/soc/usbc0/usb_host
host_chose finished!
```

3) 确认 otg_role 的模式为 usb_host 后，就可以将 OTG 转接线插入到开发板的 Micro USB 接口中，然后在 OTG 转接线的 USB 接口中插入想连接的 USB 设备即可

4) 如果接入的 USB 鼠标、键盘、U 盘或者 USB 摄像头使用正常，说明 OTG 接口的 USB Host 能正常使用

3. 22. 3. OTG 虚拟 USB 网卡测试

1) 首先需要使用 Micro USB 接口的数据线将开发板连接到电脑的 USB 接口中



2) 然后使用下面的命令将 otg_role 设置为 usb_device 模式

```
root@orangepi:~# cat /sys/devices/platform/soc/usbc0/usb_device
device_chose finished!
```

3) 然后加载 **g_ether** 内核模块

```
root@orangepi:~# modprobe g_ether
```

4) 加载完内核模块后开发板就会多出一个名为 usb0 的网络接口，首先使能这个网络接口，然后给其设置 IP 地址

```
root@orangepi:~# ifconfig usb0 up
root@orangepi:~# ifconfig usb0 192.168.10.10
root@orangepi:~# ifconfig usb0
usb0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>    mtu 1500
        inet 192.168.10.10    netmask 255.255.255.0    broadcast 192.168.10.255
        inet6 fe80::180e:b0ff:fe5a:1ee4    prefixlen 64    scopeid 0x20<link>
```

```
        ether 1a:0e:b0:5a:1e:e4  txqueuelen 1000  (Ethernet)
        RX packets 47  bytes 8223 (8.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 47  bytes 8223 (8.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

5) 然后回到 Ubuntu PC 中，可以看到多了一个名为 usb0 的网卡（也有可能为其他的名字，以实际情况为准）

```
test@test:~$ ifconfig usb0
usb0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether f6:94:ac:00:88:c6  txqueuelen 1000  (以太网)
        RX packets 12  bytes 760 (760.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 96  bytes 17733 (17.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

6) 然后在 Ubuntu PC 中给 USB 虚拟网卡分配 IP 地址，Ubuntu PC 的 IP 地址请和开发板的 IP 保持在同一网段内

```
test@test:~$ sudo ifconfig usb0 192.168.10.12
test@test:~$ ifconfig usb0
usb0 : flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.12  netmask 255.255.255.0  broadcast 192.168.10.255
        ether aa:fd:52:84:93:35  txqueuelen 1000  (以太网)
        RX packets 56  bytes 8542 (8.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 113  bytes 21351 (21.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

7) 然后就可以测试开发板和 Ubuntu PC 是否可以相互通信

```
root@orangepi:~# ping 192.168.10.12
PING 192.168.10.12 (192.168.10.12) 56(84) bytes of data.
64 bytes from 192.168.10.12: icmp_seq=1 ttl=64 time=0.376 ms
64 bytes from 192.168.10.12: icmp_seq=2 ttl=64 time=0.549 ms
64 bytes from 192.168.10.12: icmp_seq=3 ttl=64 time=0.460 ms
64 bytes from 192.168.10.12: icmp_seq=4 ttl=64 time=0.460 ms
```

```
64 bytes from 192.168.10.12: icmp_seq=5 ttl=64 time=0.493 ms
^C
```

3.22.4. OTG 虚拟串口测试

1) 首先需要使用 Micro USB 接口的数据线将开发板连接到电脑的 USB 接口中



2) 然后使用下面的命令将 otg_role 设置为 usb_device 模式

```
root@orangepi:~# cat /sys/devices/platform/soc/usbc0/usb_device
device_chose finished!
```

3) 然后加载 **g_serial** 内核模块

```
root@orangepi:~# modprobe g_serial
```

4) 加载完内核模块后开发板 linux 系统的/dev 下面就会多出一个名为 ttyGS0 的设备节点

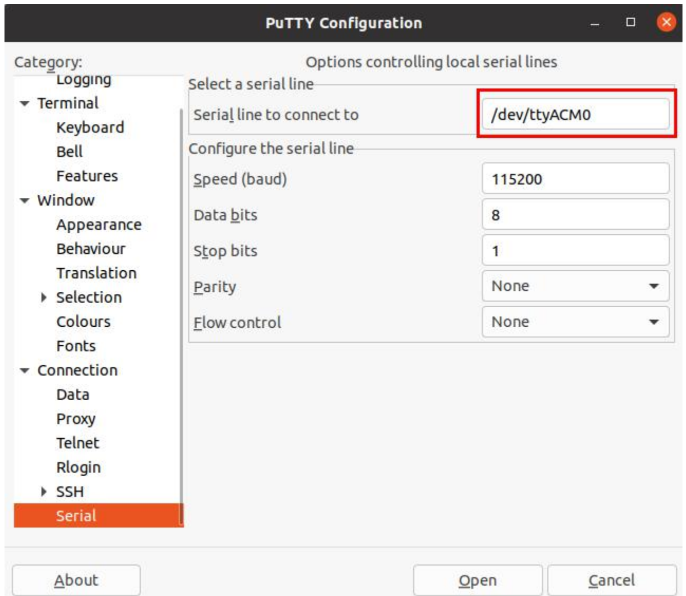
```
root@orangepi:~# ls /dev/ttyGS*
/dev/ttyGS0
```

5) 然后回到 Ubuntu PC，可以看到/dev 下面会多了一个名为 ttyACM0 的设备节点

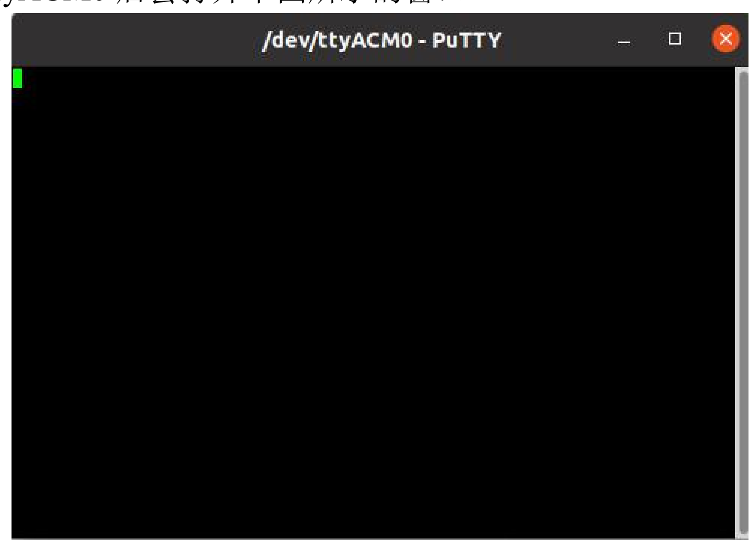
```
test@test:~$ ls /dev/ttyACM*
/dev/ttyACM0
```

6) 然后打开 Ubuntu PC 的 putty，连接 ttyACM0

```
test@test:~$ sudo putty
```



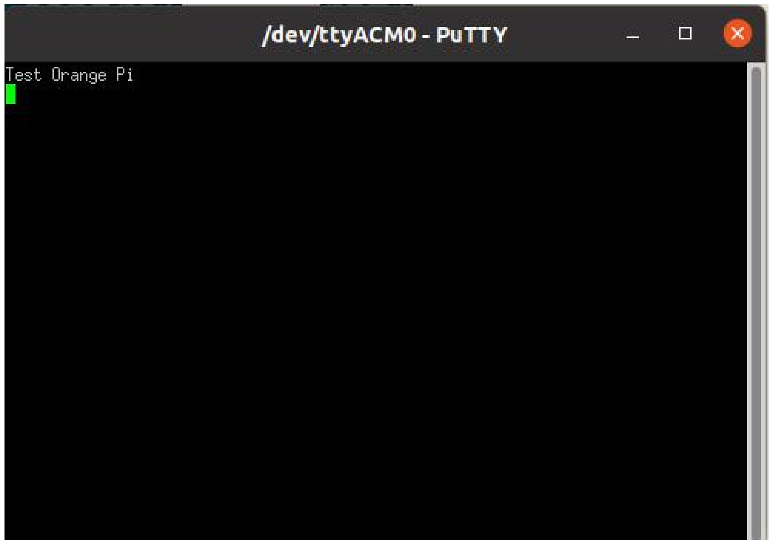
Putty 连接好 ttyACM0 后会打开下图所示的窗口



7) 然后回到开发板的 linux 系统，给/dev/ttyGS0 发送一串字符

```
root@orangepi:~# echo "Test Orange Pi" > /dev/ttyGS0
```

8) 如果一切正常，Ubuntu PC 的 Putty 就会收到开发板发送过来的字符串



3.22.5. OTG 虚拟 U 盘

1) 首先需要使用 Micro USB 接口的数据线将开发板连接到电脑的 USB 接口中



2) 然后使用下面的命令将 otg_role 设置为 usb_device 模式

```
root@orangepi:~# cat /sys/devices/platform/soc/usbc0/usb_device
device_chose finished!
```

3) 然后查看下 eMMC 的设备节点名（必须是贴有 eMMC 的 Orange Pi 3 开发板才有），使用 **ls /dev/mmcblk***命令可以列出系统中识别到的所有 MMC/SD 存储设备的节点，mmcblk0boot0 和 mmcblk0boot1 是 eMMC 中的分区，所以 eMMC 的设备节点为/dev/mmcblk0。其中/dev/mmcblk1 为 TF 卡的设备节点

如果 eMMC 中出厂烧录的安卓系统没有被覆盖，就可以看到下面所示的十几个设备节点名，如果你烧录过 linux 系统到 eMMC 中或格式化过 eMMC，那么就看不到这个多的设备节点名，可能只有一两个

```
root@orangepi3:~# ls /dev/mmcblk*
/dev/mmcblk0          /dev/mmcblk0p1     /dev/mmcblk0p12     /dev/mmcblk0p15
```

```
/dev/mmcblk0p2    /dev/mmcblk0p6    /dev/mmcblk0p9
/dev/mmcblk0boot0       /dev/mmcblk0p10       /dev/mmcblk0p13       /dev/mmcblk0p16
/dev/mmcblk0p3    /dev/mmcblk0p7    /dev/mmcblk1
/dev/mmcblk0boot1       /dev/mmcblk0p11       /dev/mmcblk0p14       /dev/mmcblk0p17
/dev/mmcblk0p5    /dev/mmcblk0p8    /dev/mmcblk1p1
```

4) 然后加载 **g_mass_storage** 内核模块，其中/dev/mmcblk0 表示将 eMMC 虚拟成 U 盘

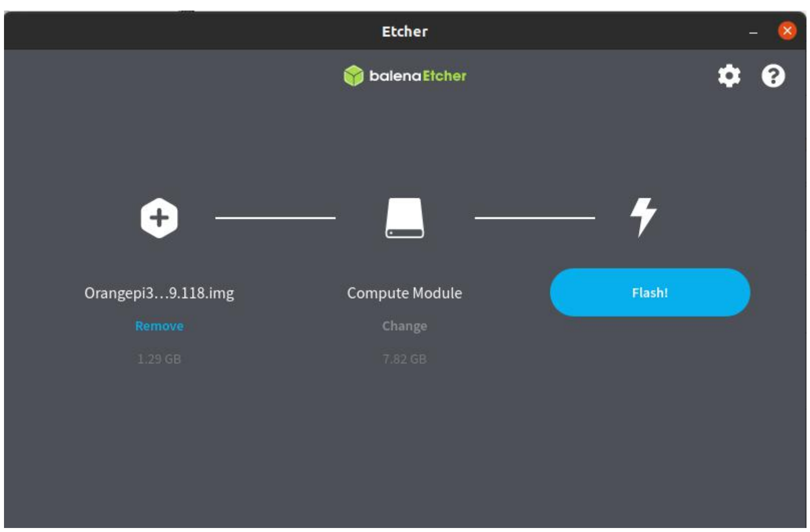
```
root@orangepi:~# modprobe g_mass_storage file=/dev/mmcblk0 stall=0
removable=1
```

5) 加载完内核模块后 Ubuntu PC 中就可以看到多了一个或多个 U 盘设备，当 eMMC 中的 Android 系统没有被破坏时，可以看到下面的多个分区设备。如果 eMMC 中烧录了 linux 镜像，那么就只会看到一个 U 盘设备

```
test@test:~$ ls /media/test/
57f8f4bc-abf4-655f-bf67-946fc0f9f25b          57f8f4bc-abf4-655f-bf67-946fc0f9f25b1
586B-07E3    system    Volumn
```

6) 此时就可以把 eMMC 当成 U 盘来用，可以对它格式化，拷贝文件等操作，当然也可以直接在 Ubuntu PC 中烧录 linux 镜像到 eMMC 中，步骤如下所示

a. 打开 balenaEtcher 软件，选择要烧录的 linux 镜像，然后选择设备为/dev/sdx（修改为 eMMC 对应的设备），从下图可以看到 eMMC 的大小为 7.82GB



b. 再点击 Flash 开始烧录 linux 镜像到 eMMC 中，和烧录 linux 镜像到 TF 卡的步骤一样

7) 烧录 linux 镜像到 eMMC 中后，就可以关闭开发板，然后拔出 TF 卡，再启动开发板，就会运行 eMMC 中的 linux 系统了

3.23. 音频测试

注意：linux5.4 系统不支持耳机播放和 MIC 录音，只有 linux4.9 系统支持

3.23.1. linux4.9 系统耳机接口播放音频测试

1) 首先在音频接口中插入耳机

2) 通过 aplay -l 命令可以查看 linux 系统支持的声卡设备，其中 **sndacx00codec** 就是耳机播放需要的声卡设备

```
root@orangepi:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: sndhdmi [sndhdmi], device 0: SUNXI-HDMIAUDIO audiohdmi-dai-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: sndacx00codec [sndacx00-codec], device 0: SUNXI-AUDIO acx00-dai-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

3) 上传需要播放的音频文件到 linux 系统的/root 文件夹中，在 Ubuntu PC 中可以使用 scp 命令上传(命令中的 IP 地址为 Orange Pi 开发板的 IP 地址)，或者使用 U 盘拷贝

```
test@test:~/AudioTest$ scp audio.wav    root@192.168.1.xx:/root   （根据实际情况修改 IP 地址和路径）
```

4) 然后使用 aplay 命令播放音频，耳机就能听到声音了

```
root@orangepi:~# aplay -D hw:1,0 audio.wav
Playing WAVE 'audio.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

3.23.2. linux4.9 系统 MIC 录音测试

1) 录音命令如下

```
root@orangepi:~# arecord -D hw:1,0 -d 5 -f cd -t wav test.wav
Recording WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

2) 录音完毕后会在当前路径下生成一个名为 test.wav 的录音文件，使用 aplay 命令可以播放 test.wav 来查看是否有声音输出，如果有声音输出，说明录音正常

```
root@orangepi:~# ls -lh
total 864K
-rw-r--r-- 1 root root 862K Dec  1 06:25 test.wav
root@orangepi:~# aplay -D hw:1,0 test.wav
```

3.23.3. HDMI 音频播放测试

1) 首先使用 HDMI 线将 Orange Pi 开发板连接到电视机上（其他的 HDMI 显示器需要确保可以播放音频）

2) 上传需要播放的音频文件到 linux 系统的/root 文件夹中，在 Ubuntu PC 中可以使用 scp 命令上传(命令中的 IP 地址为 Orange Pi 开发板的 IP 地址)，或者使用 U 盘拷贝

```
test@test:~/AudioTest$ scp audio.wav  root@192.168.1.xx:/root （根据实际情况修改 IP 地址和路径）
```

3) HDMI 音频播放无需其他设置，直接使用 aplay 命令播放即可

```
root@orangepi:~# aplay -D hw:0,0 audio.wav
```

3.24. 红外接收测试

1) 安装 ir-keytable 红外测试软件

```
root@orangepi:~# apt update
root@orangepi:~# apt-get install ir-keytable
```

2) 然后执行 ir-keytable 可以查看红外设备的信息

a. linux4.9 系统输出如下所示

```
root@orangepi:~# ir-keytable
Found /sys/class/rc/rc0/ (/dev/input/event4) with:
        Driver sunxi-rc-recv, table rc_map_sunxi
```

```
        Supported protocols: nec
        Enabled protocols: nec
        Name: sunxi-ir
        bus: 25, vendor/product: 0001:0001, version: 0x0100
        Repeat delay = 500 ms, repeat period = 125 ms
```

b. linux5.4 系统输出如下所示

```
root@orangepi:~# ir-keytable
Found /sys/class/rc/rc0/ (/dev/input/event1) with:
        Name: sunxi-ir
        Driver: sunxi-ir, table: rc-empty
        LIRC device: /dev/lirc0
        Attached BPF protocols: Operation not supported
        Supported kernel protocols: other lirc rc-5 rc-5-sz jvc sony nec sanyo mce_kb
        Enabled kernel protocols: lirc
        bus: 25, vendor/product: 0001:0001, version: 0x0100
        Repeat delay = 500 ms, repeat period = 125 ms
```

3) 测试红外接收功能前需要准备一个红外遥控器



4) 然后在终端中输入 ir-keytable -t 命令，再使用红外遥控器对着 Orange Pi 开发板的红外接收头按下按键就能在终端中看到接收到的按键编码了

a. linux4.9 系统输出如下所示

```
root@orangepi:/# ir-keytable -t
Testing events. Please, press CTRL-C to abort.
1598339152.260376: event type EV_MSC(0x04): scancode = 0xfb0413
1598339152.260376: event type EV_SYN(0x00).
1598339152.914715: event type EV_MSC(0x04): scancode = 0xfb0410
```

b. linux5.4 系统输出如下所示

```
root@orangepi:~# ir-keytable -c -p NEC -t
Old keytable cleared
Protocols changed to nec
Testing events. Please, press CTRL-C to abort.
3161.772262: lirc protocol(nec): scancode = 0x45c
3161.772323: event type EV_MSC(0x04): scancode = 0x45c
3161.772323: event type EV_SYN(0x00).
```

3.25. 温度传感器

1) H6 总共有 2 个温度传感器，查看温度的命令如下

a. sensor0：CPU

```
root@orangepi:~# cat /sys/class/thermal/thermal_zone0/type
cpu_thermal_zone
root@orangepi:~# cat /sys/class/thermal/thermal_zone0/temp
57734
```

b. sensor1：GPU

```
root@orangepi:~# cat /sys/class/thermal/thermal_zone1/type
gpu_thermal_zone
root@orangepi:~# cat /sys/class/thermal/thermal_zone1/temp
57410
```

3.26. 安装 Docker 的方法

1) 先卸载可能存在的旧版本 docker

```
root@orangepi:~# apt remove docker docker-engine docker-ce docker.io
```

2) 然后安装下面的软件包

```
root@orangepi:~# apt update
root@orangepi:~# apt install -y apt-transport-https ca-certificates curl
  software-properties-common          (这是一条命令)
```

3) 再添加阿里云 docker 的密钥

```
root@orangepi:~# curl -fsSL http://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg
 | sudo apt-key add  -                    (这是一条命令)
```

4) 在 ubuntu 的系统源中添加对应的 docker 源

```
root@orangepi:~# add-apt-repository "deb [arch=arm64]
    https://mirrors.aliyun.com/docker-ce/linux/ubuntu $(lsb_release -cs) stable"
  (这是一条命令)
```

5) 安装最新版本的 docker-ce

```
root@orangepi:~# apt update
root@orangepi:~# apt install docker-ce
```

6) 验证 docker 的状态

```
root@orangepi:~# systemctl status docker
● docker.service - Docker Application Container Engine
    Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
    Active: active (running) since Mon 2020-08-24 10:29:22 UTC; 26min ago
      Docs: https://docs.docker.com
  Main PID: 3145 (dockerd)
     Tasks: 15
    CGroup: /system.slice/docker.service
            └─3145 /usr/bin/dockerd -H fd://
--containerd=/run/containerd/containerd.sock
```

7) 测试 docker

```
root@orangepi:~# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
256ab8fe8778: Pull complete
Digest:
sha256:7f0a9f93b4aa3022c3a4c147a449ef11e0941a1fd0bf4a8e6c9408b2600777c5
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

3.27. 26 Pin 接口引脚说明

1) Orange Pi 3 开发板 26 pin 接口引脚的顺序请参考下图



2) Orange Pi 3 开发板 26 pin 接口引脚的功能如下表所示

GPIO序号	GPIO	功能	引脚	引脚	功能	GPIO	GPIO序号
		3.3V	1	2	5V		
122	PD26	TWI0-SDA	3	4	5V		
121	PD25	TWI0-SCK	5	6	GND		
118	PD22	PWM0	7	8	S-UART_TX	PL2	354
		GND	9	10	S-UART_RX	PL3	355
120	PD24	UART3_RX	11	12	PD18	PD18	114
119	PD23	UART3_TX	13	14	GND		
362	PL10	PL10	15	16	PD15	PD15	111
		3.3V	17	18	PD16	PD16	112
229	PH5	SPI1_MOSI	19	20	GND		
230	PH6	SPI1_MISO	21	22	PD21	PD21	117
228	PH4	SPI1_CLK	23	24	SPI1_CS	PH3	227
		GND	25	26	PL8	PL8	360

3.28. 安装 wiringOP 的方法

1) 下载 wiringOP 的代码

```
root@orangepi:~# apt update
root@orangepi:~# apt install git
root@orangepi:~# git clone https://github.com/orangepi-xunlong/wiringOP
```

2) 编译 wiringOP

```
root@orangepi:~# cd wiringOP
root@orangepi:~/wiringOP# ./build clean
root@orangepi:~/wiringOP# ./build
```

3) 测试 gpio readall 命令的输出如下，其中 1 到 26 号物理引脚与开发板上的 26 Pin 引脚是一一对应的

```
root@orangepi3:~/wiringOP# gpio readall
 +------+-----+----------+------+---+   OPi 3  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     3.3V |      |   |  1 || 2  |   |      | 5V       |     |      |
 |  122 |   0 |    SDA.0 |  OFF | 0 |  3 || 4  |   |      | 5V       |     |      |
 |  121 |   1 |    SCL.0 |  OFF | 0 |  5 || 6  |   |      | GND      |     |      |
 |  118 |   2 |    PWM.0 |  OFF | 0 |  7 || 8  | 0 | OFF  | PL02     | 3   | 354  |
 |      |     |      GND |      |   |  9 || 10 | 0 | OFF  | PL03     | 4   | 355  |
 |  120 |   5 |    RXD.3 | ALT4 | 0 | 11 || 12 | 0 | OFF  | PD18     | 6   | 114  |
 |  119 |   7 |    TXD.3 | ALT4 | 0 | 13 || 14 |   |      | GND      |     |      |
 |  362 |   8 |     PL10 |  OFF | 0 | 15 || 16 | 0 | OFF  | PD15     | 9   | 111  |
 |      |     |     3.3V |      |   | 17 || 18 | 0 | OFF  | PD16     | 10  | 112  |
 |  229 |  11 |   MOSI.1 | ALT2 | 0 | 19 || 20 |   |      | GND      |     |      |
 |  230 |  12 |   MISO.1 | ALT2 | 0 | 21 || 22 | 0 | OFF  | PD21     | 13  | 117  |
 |  228 |  14 |   SCLK.1 | ALT2 | 0 | 23 || 24 | 0 | ALT2 | CE.1     | 15  | 227  |
 |      |     |      GND |      |   | 25 || 26 | 0 | OFF  | PL08     | 16  | 360  |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+   OPi 3  +---+------+----------+-----+------+
root@orangepi3:~/wiringOP#
```

3.29. 26pin 接口 GPIO、I2C、UART、SPI 测试

wiringOP 已适配 Orange Pi 3 开发板，使用 wiringOP 可以测试 GPIO、I2C、UART 和 SPI 的功能

3.29.1. 26pin GPIO 口测试

1) 下面以 7 号引脚——对应 GPIO 为 PD22 ——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的高低电平

```
root@orangepi3:~/wiringOP# gpio readall
 +------+-----+----------+------+---+   OPi 3  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     3.3V |      |   |  1 || 2  |   |      | 5V       |     |      |
 |  122 |   0 |    SDA.0 |  OFF | 0 |  3 || 4  |   |      | 5V       |     |      |
 |  121 |   1 |    SCL.0 |  OFF | 0 |  5 || 6  |   |      | GND      |     |      |
 |  118 |   2 |    PWM.0 |  OFF | 0 |  7 || 8  | 0 | OFF  | PL02     | 3   | 354  |
 |      |     |      GND |      |   |  9 || 10 | 0 | OFF  | PL03     | 4   | 355  |
 |  120 |   5 |    RXD.3 | ALT4 | 0 | 11 || 12 | 0 | OFF  | PD18     | 6   | 114  |
```

2) 首先设置 GPIO 口为输出模式，其中第三个参数需要输入引脚对应的 wPi 的序号

```
root@orangepi:~/wiringOP# gpio mode 2 out
```

使用 gpio readall 可以看到 7 号引脚的模式(mode)变为了 out

```
root@orangepi3:~/wiringOP# gpio readall
 +------+-----+----------+------+---+   OPi 3  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     3.3V |      |   |  1 || 2  |   |      | 5V       |     |      |
 |  122 |   0 |    SDA.0 |  OFF | 0 |  3 || 4  |   |      | 5V       |     |      |
 |  121 |   1 |    SCL.0 |  OFF | 0 |  5 || 6  |   |      | GND      |     |      |
 |  118 |   2 |    PWM.0 |  OUT | 0 |  7 || 8  | 0 | OFF  | PL02     | 3   | 354  |
 |      |     |      GND |      |   |  9 || 10 | 0 | OFF  | PL03     | 4   | 355  |
```

3) 然后设置 GPIO 口输出低电平，设置完后可以使用万用表测量引脚的电压的数值，如果为 0v，说明设置低电平成功

```
root@orangepi:~/wiringOP# gpio write 2 0
```

使用 gpio readall 可以看到 7 号引脚的值(V)变为了 0

```
root@orangepi3:~/wiringOP# gpio readall
 +------+-----+----------+------+---+   OPi 3  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     3.3V |      |   |  1 || 2  |   |      | 5V       |     |      |
 |  122 |   0 |    SDA.0 |  OFF | 0 |  3 || 4  |   |      | 5V       |     |      |
 |  121 |   1 |    SCL.0 |  OFF | 0 |  5 || 6  |   |      | GND      |     |      |
 |  118 |   2 |    PWM.0 |  OUT | 0 |  7 || 8  | 0 | OFF  | PL02     | 3   | 354  |
 |      |     |      GND |      |   |  9 || 10 | 0 | OFF  | PL03     | 4   | 355  |
```

4) 然后设置 GPIO 口输出高电平，设置完后可以使用万用表测量引脚的电压的数值，如果为 3.3v，说明设置高电平成功

```
root@orangepi:~/wiringOP# gpio write 2 1
```

使用 gpio readall 可以看到 7 号引脚的值(V)变为了 1

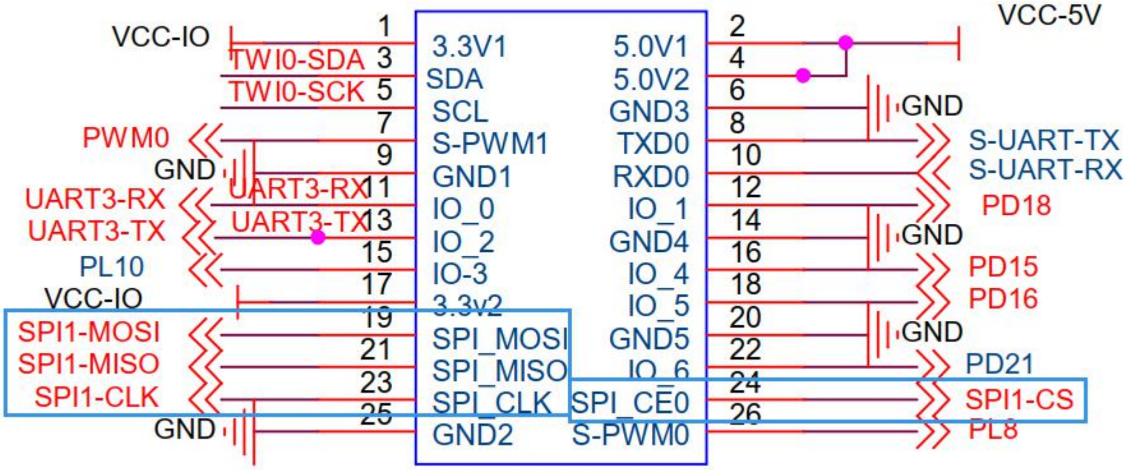
```
root@orangepi3:~/wiringOP# gpio readall
 +------+-----+----------+------+---+   OPi 3  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     3.3V |      |   |  1 || 2  |   |      | 5V       |     |      |
 |  122 |   0 |    SDA.0 |  OFF | 0 |  3 || 4  |   |      | 5V       |     |      |
 |  121 |   1 |    SCL.0 |  OFF | 0 |  5 || 6  |   |      | GND      |     |      |
 |  118 |   2 |    PWM.0 |  OUT | 1 |  7 || 8  | 0 | OFF  | PL02     | 3   | 354  |
 |      |     |      GND |      |   |  9 || 10 | 0 | OFF  | PL03     | 4   | 355  |
```

5) 其他引脚的设置方法类似，只需修改 wPi 的序号为引脚对应的序号即可

3. 29. 2. 26pin SPI 测试

1) linux5.4 系统在 dts 中默认关闭了 26pin 中的 spi 控制器，如果需要使用 spi，首先需要先打开 spi 的配置，**linux4.9 系统默认都是打开的**，不需要额外的配置，linux5.4 系统 spi 的打开方法如下所示：

a. 由 26pin 的原理图可知，开发板可用的 spi 为 spi1



b. 然后在 /boot/orangepiEnv.txt 中设置 overlays=spi-spidev1

```
overlays=spi-spidev1
```

c. 然后重启系统，启动时，在 u-boot 的启动 log 中可以看到 SPI DT overlays 的配置输出

```
Applying kernel provided DT overlay sun50i-h6-spi-spidev1.dtbo
4191 bytes read in 6 ms (681.6 KiB/s)
Applying kernel provided DT fixup script (sun50i-h6-fixup.scr)
```

d. 系统启动后，在/dev 下如果可以看到 SPI 的设备节点就说明配置正确

```
root@orangepi:~# ls /dev/spi*
/dev/spidev1.0
```

2) 再在 wiringOP 的 examples 中编译 spidev_test 测试程序

```
root@orangepi:~/wiringOP/examples# make spidev_test
[CC] spidev_test.c
[link]
```

3) 先不短接 SPI1 的 mosi 和 miso 两个引脚，运行 spidev_test 的输出结果如下所示，可以看到 TX 和 RX 的数据不一致

```
root@orangepi:~/wiringOP/examples# ./spidev_test -v -D /dev/spidev1.0
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D  | ......@....▒.................▒.
RX | FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF  | ................................
```

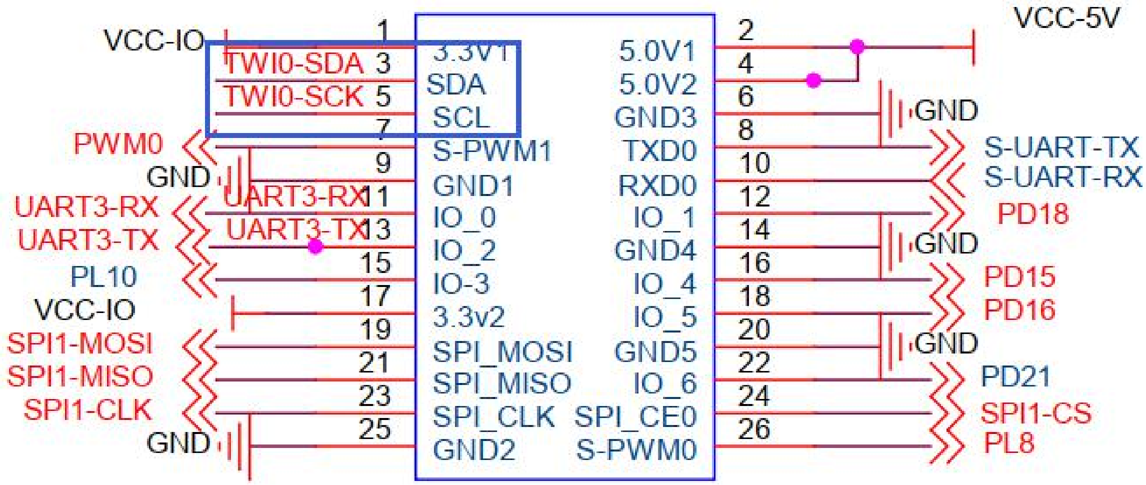
4) 然后短接 SPI1 的 mosi（26pin 接口中的第 19 号引脚）和 miso（26pin 接口中的第 21 号引脚）两个引脚再运行 spidev_test 的输出如下，可以看到发送和接收的数据一样

```
root@orangepi:~/wiringOP/examples# ./spidev_test -v -D /dev/spidev1.0
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D  | ......@....▒.................▒.
RX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D  | ......@....▒.................▒.
```

3. 29. 3. 26pin I2C 测试

1) linux5.4 系统在 dts 中默认关闭了 26pin 中的 i2c 控制器，如果需要使用 i2c，首先需要先打开 i2c 的配置，**linux4.9 系统默认都是打开的**，不需要额外的配置，linux5.4 系统 i2c 的打开方法如下所示：

a. 由 26pin 的原理图可知，开发板可用的 i2c 为 i2c0



b. 然后在 /boot/orangepiEnv.txt 中设置 overlays=i2c0 就可以打开 i2c0 的配置

```
overlays=i2c0
```

c. 然后重启系统，启动时，在 u-boot 的启动 log 中可以看到 I2C DT overlays 的配置输出

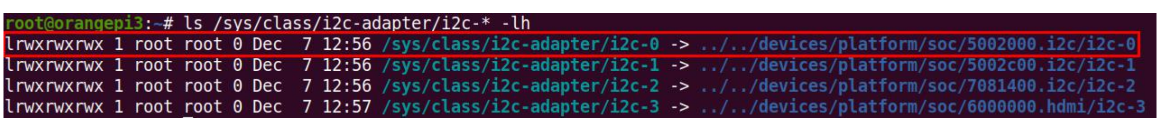
```
Applying kernel provided DT overlay sun50i-h6-i2c0.dtbo
4191 bytes read in 5 ms (818.4 KiB/s)
Applying kernel provided DT fixup script (sun50i-h6-fixup.scr)
```

d. 系统启动后，在/dev 下如果多了个 i2c 的设备节点就说明配置正确

```
root@orangepi:~# ls /dev/i2c*
/dev/i2c-0    /dev/i2c-1    /dev/i2c-2    /dev/i2c-3
```

e. 不同的 i2c 设备节点的对应关系如下所示，其中

a) 26pin 中的 i2c0 对应/dev/i2c-0



```
root@orangepi3:~# ls /sys/class/i2c-adapter/i2c-* -lh
lrwxrwxrwx 1 root root 0 Dec  7 12:56 /sys/class/i2c-adapter/i2c-0 -> ../../devices/platform/soc/5002000.i2c/i2c-0
lrwxrwxrwx 1 root root 0 Dec  7 12:56 /sys/class/i2c-adapter/i2c-1 -> ../../devices/platform/soc/5002c00.i2c/i2c-1
lrwxrwxrwx 1 root root 0 Dec  7 12:56 /sys/class/i2c-adapter/i2c-2 -> ../../devices/platform/soc/7081400.i2c/i2c-2
lrwxrwxrwx 1 root root 0 Dec  7 12:57 /sys/class/i2c-adapter/i2c-3 -> ../../devices/platform/soc/6000000.hdmi/i2c-3
```

2) 然后开始测试 i2c，首先安装 i2c-tools

```
root@orangepi:~# apt update
root@orangepi:~# apt install i2c-tools
```

3) 然后在 26pin 接头的 i2c0 引脚上接一个 i2c 设备

	i2c0
sda 引脚	对应 3 号引脚
sck 引脚	对应 5 号引脚

vcc 引脚	对应 1 号引脚
gnd 引脚	对应 6 号引脚

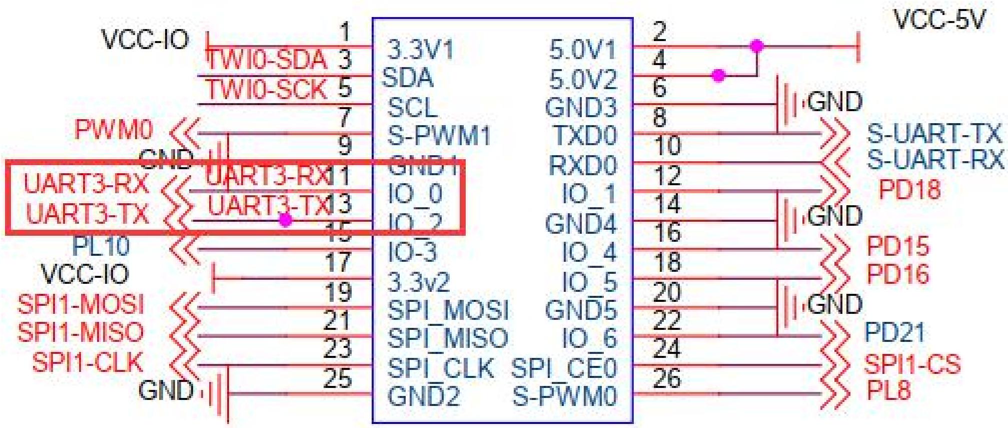
4) 然后使用 i2cdetect -y 0 命令如果能检测到连接的 i2c 设备的地址，就说明 i2c 能正常使用

```
root@orangepi:~# i2cdetect -y 3
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

3. 29. 4. 26pin 的 UART 测试

1) linux5.4 系统在 dts 中默认关闭了 26pin 中的 uart 控制器，如果需要使用 uart，首先需要先打开 uart 的配置，**linux4.9 系统默认都是打开的**，不需要额外的配置，linux5.4 系统 uart 的打开方法如下所示：

a. 由 26pin 的原理图可知，开发板可用的 uart 为 uart3



b. 然后在 /boot/orangepiEnv.txt 中设置 overlays=uart3 就可以打开 uart3 的配置

```
overlays=uart3
```

c. 然后重启系统，启动时，在 u-boot 的启动 log 中可以看到 UART 相关的 DT overlays 的配置输出

```
Applying kernel provided DT overlay sun50i-h6-uart3.dtbo
```

```
268 bytes read in 4 ms (65.4 KiB/s)
Applying kernel provided DT fixup script (sun50i-h6-fixup.scr)
```

d. 系统启动后，在/sys/class/tty 下可以看到 ttyS3 的信息，26pin 中的 uart3 对应/dev/ttyS3

```
root@orangepi3:~# ls /sys/class/tty/ttyS* -lh
lrwxrwxrwx 1 root root 0 Jan  1  1970 /sys/class/tty/ttyS0 -> ../../devices/platform/soc/5000000.serial/tty/ttyS0
lrwxrwxrwx 1 root root 0 Dec  7 12:38 /sys/class/tty/ttyS1 -> ../../devices/platform/soc/5000400.serial/tty/ttyS1
lrwxrwxrwx 1 root root 0 Jan  1  1970 /sys/class/tty/ttyS2 -> ../../devices/platform/serial8250/tty/ttyS2
lrwxrwxrwx 1 root root 0 Dec  7 12:38 /sys/class/tty/ttyS3 -> ../../devices/platform/soc/5000c00.serial/tty/ttyS3
lrwxrwxrwx 1 root root 0 Jan  1  1970 /sys/class/tty/ttyS4 -> ../../devices/platform/serial8250/tty/ttyS4
lrwxrwxrwx 1 root root 0 Jan  1  1970 /sys/class/tty/ttyS5 -> ../../devices/platform/serial8250/tty/ttyS5
lrwxrwxrwx 1 root root 0 Jan  1  1970 /sys/class/tty/ttyS6 -> ../../devices/platform/serial8250/tty/ttyS6
lrwxrwxrwx 1 root root 0 Jan  1  1970 /sys/class/tty/ttyS7 -> ../../devices/platform/serial8250/tty/ttyS7
```

2) 然后开始测试 uart 接口，先使用杜邦线短接要测试的 uart3 接口的 rx 和 tx

	uart3
tx 引脚	对应 13 号引脚
rx 引脚	对应 11 号引脚

3) 然后修改 wiringOP 中串口测试程序 serialTest 打开的串口设备节点名为**/dev/ttyS3**

```
root@orangepi3:~/wiringOP/examples# vim serialTest.c
```

```
int main ()
{
  int fd ;
  int count ;
  unsigned int nextTime ;

  if ((fd = serialOpen ("/dev/ttyS3", 115200)) < 0)
  {
    fprintf (stderr, "Unable to open serial device: %s\n", strerror (errno)) ;
    return 1 ;
  }
```

4) 再编译 wiringOP 中的串口测试程序 serialTest

```
root@orangepi:~/wiringOP/examples# make serialTest
[CC] serialTest.c
[link]
root@orangepi:~/wiringOP/examples#
```

5) 最后运行 serialTest，如果能看到下面的打印，说明串口通信正常

```
root@orangepi:~/wiringOP/examples# ./serialTest

Out:     0:   ->     0
```

```
Out:    1:  ->   1
Out:    2:  ->   2
Out:    3:  ->   3
Out:    4:  ->   4
Out:    5:  ->   5
Out:    6:  ->   6
Out:    7:  ->   7
Out:    8:  ->   8^C
```

3. 30. I2C 接口的 0.96 寸 OLED 模块使用方法

1) 香橙派的 0.96 寸 OLED 模块如下图所示，其 7 位的 i2c 从机地址为 0x3c



2) 先将 0.96 寸 OLED 模块通过杜邦线连接到 Orange Pi 开发板的 26pin 接口上，接线方式如下所示

OLED 模块的引脚	描述	开发板 26pin 接口对应引脚
GND	电源地	6 号引脚
VCC	5V	2 号引脚
SCL	I2C 时钟线	5 号引脚
SDA	I2C 数据线	3 号引脚
RST	接 3.3V	1 号引脚
DC	接 GND	9 号引脚
CS	接 GND	25 号引脚



3) 将 OLED 模块连接到开发板后，先使用 i2c-tools 工具检查下是否能扫描到 OLED 模块的地址

```
root@orangepi:~# apt update
root@orangepi:~# apt install i2c-tools
root@orangepi:~# i2cdetect -y 0
```

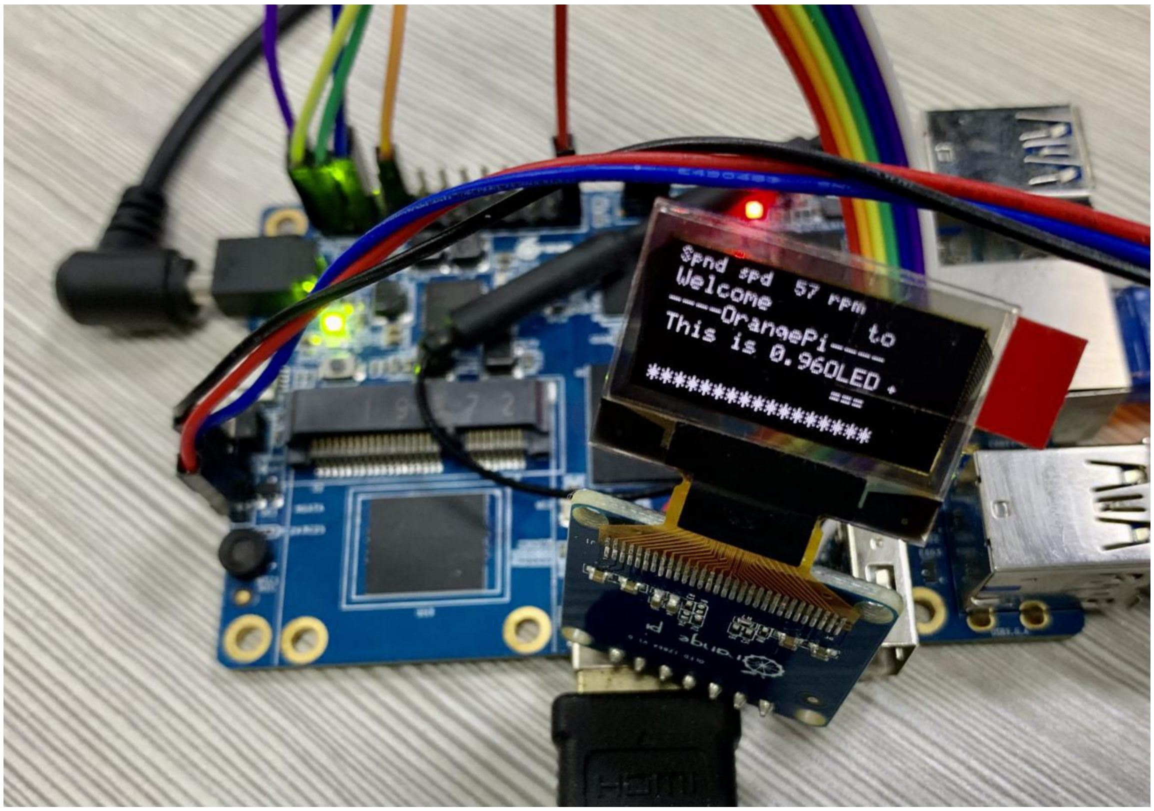
```
root@orangepi3:~# i2cdetect -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- 3c -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
root@orangepi3:~#
```

4) 然后就可以使用 wiringOP 中的 **oled_demo** 来测试 OLED 模块，测试步骤如下所示

```
root@orangepi:~# git clone https://github.com/orangepi-xunlong/wiringOP
root@orangepi:~# cd wiringOP
root@orangepi:~/wiringOP# ./build clean && ./build
```

```
root@orangepi:~/wiringOP# cd examples
root@orangepi:~/wiringOP/examples# make oled_demo
root@orangepi:~/wiringOP/examples# ./oled_demo /dev/i2c-0
---------start--------
----------end---------
```

5) 运行 oled_demo 后，在 OLED 屏幕上就能看到下面的输出



3. 31. SPI LCD 显示屏使用方法

注意：此方法只适用于 linux4.9 内核的系统，linux5.4 内核的系统无法使用

3. 31. 1. 2.4 寸 SPI LCD 显示屏

1) 测试的 LCD 显示屏详情页链接如下

http://www.lcdwiki.com/2.4inch_SPI_Module_ILI9341_SKU:MSP2402

2) LCD 显示屏和开发板的接线方式如下所示

TFT SPI 模块引脚	开发板 26pin 对应的引脚	GPIO -- GPIO num

VCC	1 号引脚	
GND	6 号引脚	
CS	24 号引脚	
RESET	12 号引脚	PD18 -- 114
D/C	16 号引脚	PD15 -- 111
SDI(MOSI)	19 号引脚	
SCK	23 号引脚	
LED	18 号引脚	PD16 -- 112
SDO(MISO)	21 引脚	

3) 将显示屏接到开发板后，再使用下面的命令加载 **fbtft_device** 内核模块

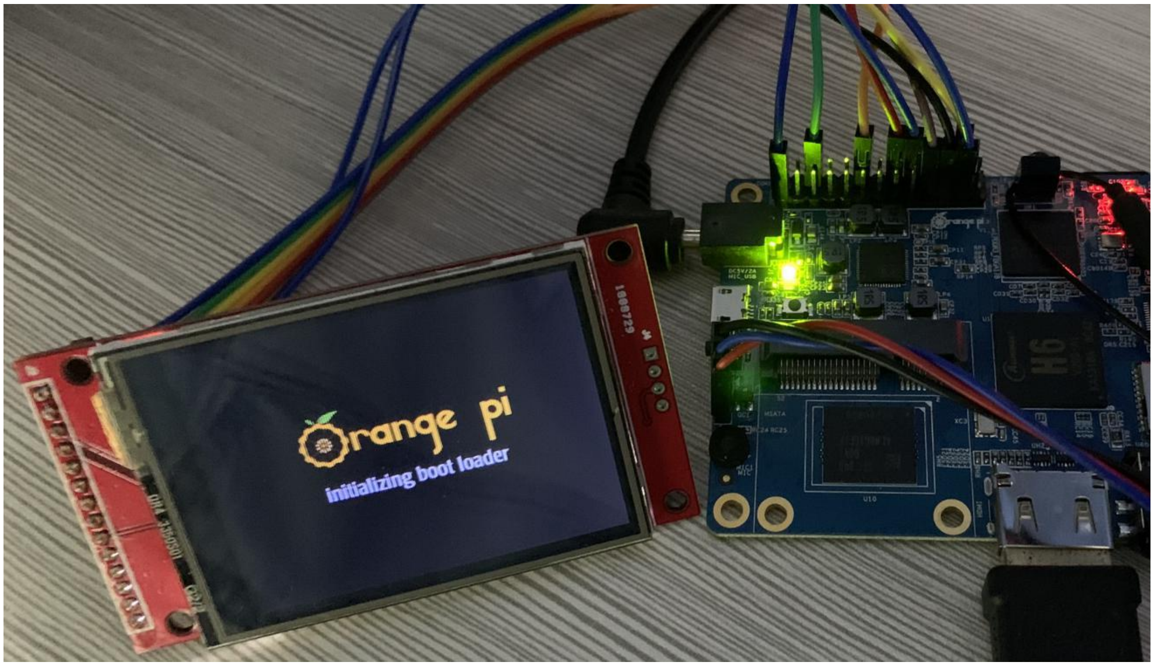
```
root@orangepi:~# modprobe fbtft_device custom name=fb_ili9341  busnum=1 cs=0
gpios=reset:114,dc:111,led:112 rotate=90 speed=65000000 bgr=1 txbuflen=65536
```

4) **fbtft_device** 内核模块加载时 dmesg 命令正确的输出 log 如下所示，而且由 log 可以知道 LCD 显示屏使用的 framebuffer 为 **fb8**

```
root@orangepi:~# dmesg | tail
[  191.788887] spidev spi1.0: dh2228fv spi1.0 1200kHz 8 bits mode=0x00
[  191.789343] spidev spi1.0: Deleting spi1.0
[  191.793283] fbtft_device: GPIOS used by 'fb_ili9341':
[  191.793301] fbtft_device: 'reset' = GPIO114
[  191.793313] fbtft_device: 'dc' = GPIO111
[  191.793324] fbtft_device: 'led' = GPIO112
[  191.793351] spi spi1.0: fb_ili9341 spi1.0 65000kHz 8 bits mode=0x00
[  191.807788] fb_ili9341: module is from the staging directory, the quality is unknown,
you have been warned.
[  192.083355] graphics fb8: fb_ili9341 frame buffer, 320x240, 150 KiB video memory,
64 KiB buffer memory, fps=20, spi1.0 at 65 MHz
```

5) 然后使用下面的命令就可以在 LCD 显示屏上显示 Orange Pi 的 logo 图片

```
root@orangepi:~# apt update
root@orangepi:~# apt -y install fbi
root@orangepi:~# fbi -vt 1 -noverbose -d /dev/fb8 /boot/boot.bmp
```



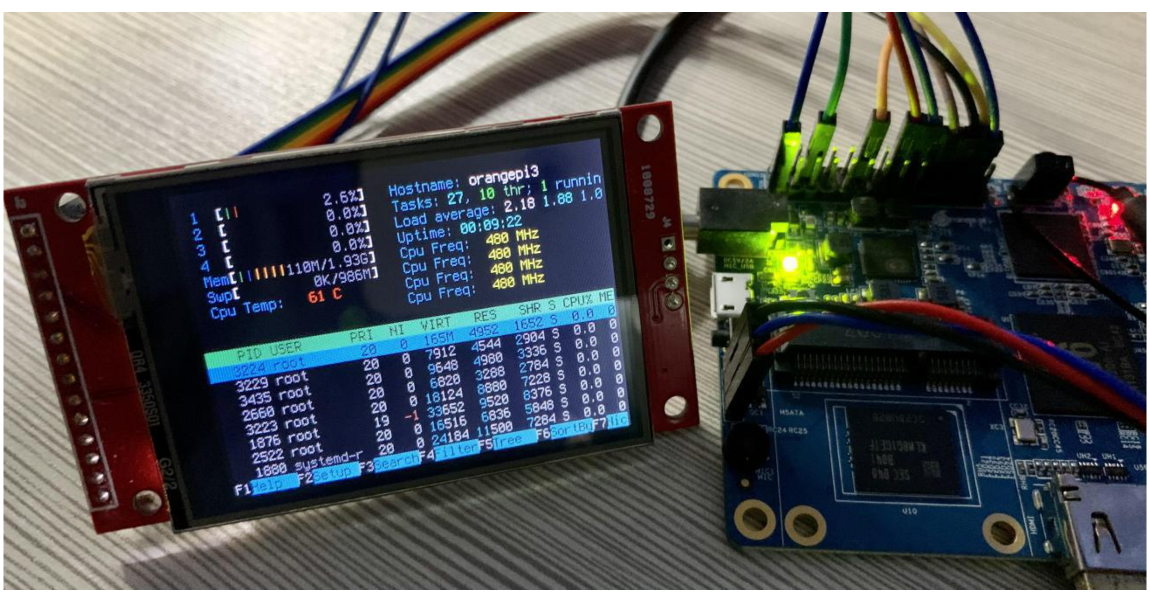
6) 还可以将 tty1 的输出映射到 LCD 显示屏的 fb 设备——**fb8**，映射完后，HDMI 就不会再有图像输出了

```
root@orangepi:~# con2fbmap 1 8
```

如果要切换回 HDMI 显示，请使用下面的命令

```
root@orangepi:~# con2fbmap 1 0
```

下面是运行 htop 命令的输出

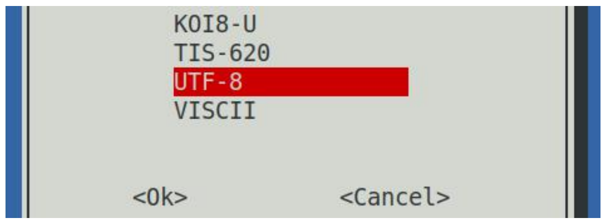


7) 由于默认的终端字体太大，导致显示屏无法显示太多的内容，可以通过下面的方法来缩小终端的字体

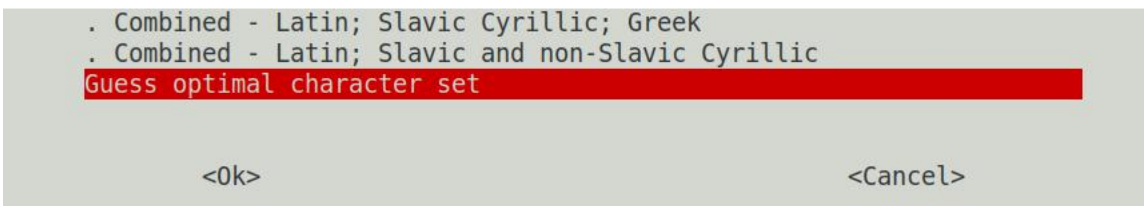
a. 首先运行 **dpkg-reconfigure console-setup**

```
root@orangepi:~# apt-get update
root@orangepi:~# apt-get install kbd
root@orangepi:~# dpkg-reconfigure console-setup
```

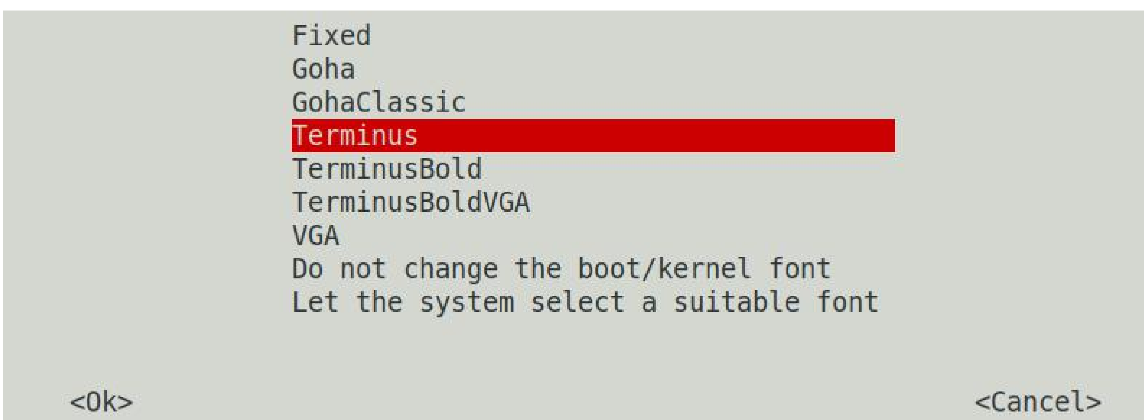
b. 终端编码选择 **UTF-8**



c. 然后选择 **Guess optimal character set**



d. 然后选择 **Terminus**



e. 最后选择字体大小为 6x12

```
6x12 (framebuffer only)
8x14
8x16
10x20 (framebuffer only)
11x22 (framebuffer only)
12x24 (framebuffer only)
14x28 (framebuffer only)
16x32 (framebuffer only)

<Ok>                                    <Cancel>
```

f. 设置完后就能看到 LCD 显示屏上的字体变小了

8) 设置系统启动自动加载 fbtft_device 模块的方法

a. 新建**/etc/modules-load.d/fbtft.con** 配置文件，文件内容如下所示

```
root@orangepi:~# cat /etc/modules-load.d/fbtft.conf
fbtft_device
```

b. 新建**/etc/modprobe.d/fbtft.conf** 配置文件，文件内容如下所示

```
root@orangepi:~# cat /etc/modprobe.d/fbtft.conf
options    fbtft_device custom name=fb_ili9341    busnum=1 cs=0
gpios=reset:114,dc:111,led:112 rotate=90 speed=65000000 bgr=1 txbuflen=65536
```

c. 然后重启 linux 系统就能看到 fbtft_device 相关的内核模块都已自动加载

9) 如果希望 linux 系统启动后自动将 console 映射到 LCD 显示屏，请在**/boot/orangepiEnv.txt** 中加入下面的配置，然后重启系统就能看到 LCD 显示屏有输出了

```
root@orangepi:~# cat /boot/orangepiEnv.txt | grep "fbcon"
extraargs=cma=25M fbcon=map:8
```

3.31.2. 3.2 寸 RPi SPI LCD 显示屏

1) 测试的 LCD 显示屏详情页链接如下

http://www.lcdwiki.com/3.2inch_RPi_Display

2) LCD 显示屏和开发板接线方式如下所示



3) 将 LCD 显示屏接到开发板后，再使用下面的命令加载 **fbtft_device** 内核模块

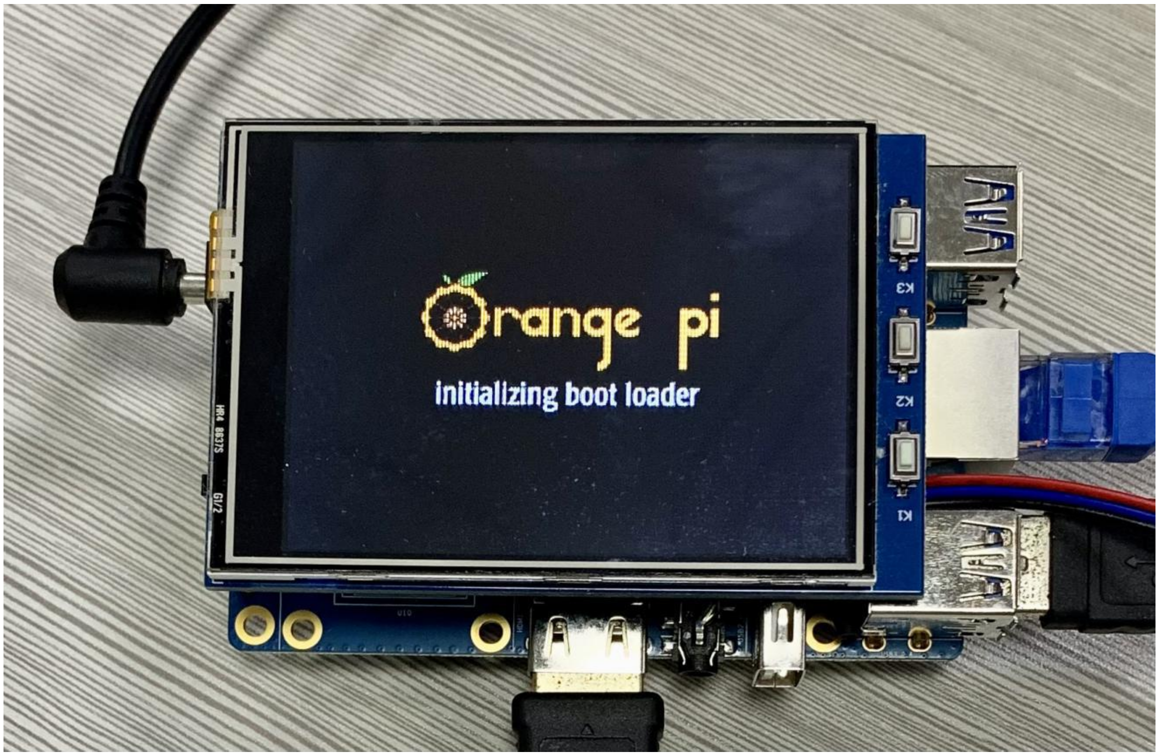
```
root@orangepi:~# modprobe fbtft_device custom name=fb_ili9341 busnum=1 cs=0 gpios=reset:119,dc:362 rotate=90 speed=65000000 bgr=1 txbuflen=65536
```

4) **fbtft_device** 内核模块加载时 dmesg 命令正确的输出 log 如下所示，而且由 log 可以知道 LCD 屏幕使用的 framebuffer 为 **fb8**

```
root@orangepi:~# dmesg | tail
[   97.884472] spidev spi1.0: dh2228fv spi1.0 1200kHz 8 bits mode=0x00
[   97.884933] spidev spi1.0: Deleting spi1.0
[   97.888354] fbtft_device: GPIOS used by 'fb_ili9341':
[   97.888373] fbtft_device: 'reset' = GPIO119
[   97.888385] fbtft_device: 'dc' = GPIO362
[   97.889430] spi spi1.0: fb_ili9341 spi1.0 65000kHz 8 bits mode=0x00
[   97.907191] fb_ili9341: module is from the staging directory, the quality is unknown, you have been warned.
[   98.181286] Console: switching to colour frame buffer device 40x30
[   98.181938] graphics fb8: fb_ili9341 frame buffer, 320x240, 150 KiB video memory, 64 KiB buffer memory, fps=20, spi1.0 at 65 MHz
```

5) 然后使用下面的命令就可以在 LCD 屏幕上显示 Orange Pi 的 logo 图片

```
root@orangepi:~# apt update
root@orangepi:~# apt -y install fbi
root@orangepi:~# fbi -vt 1 -noverbose -d /dev/fb8 /boot/boot.bmp
```



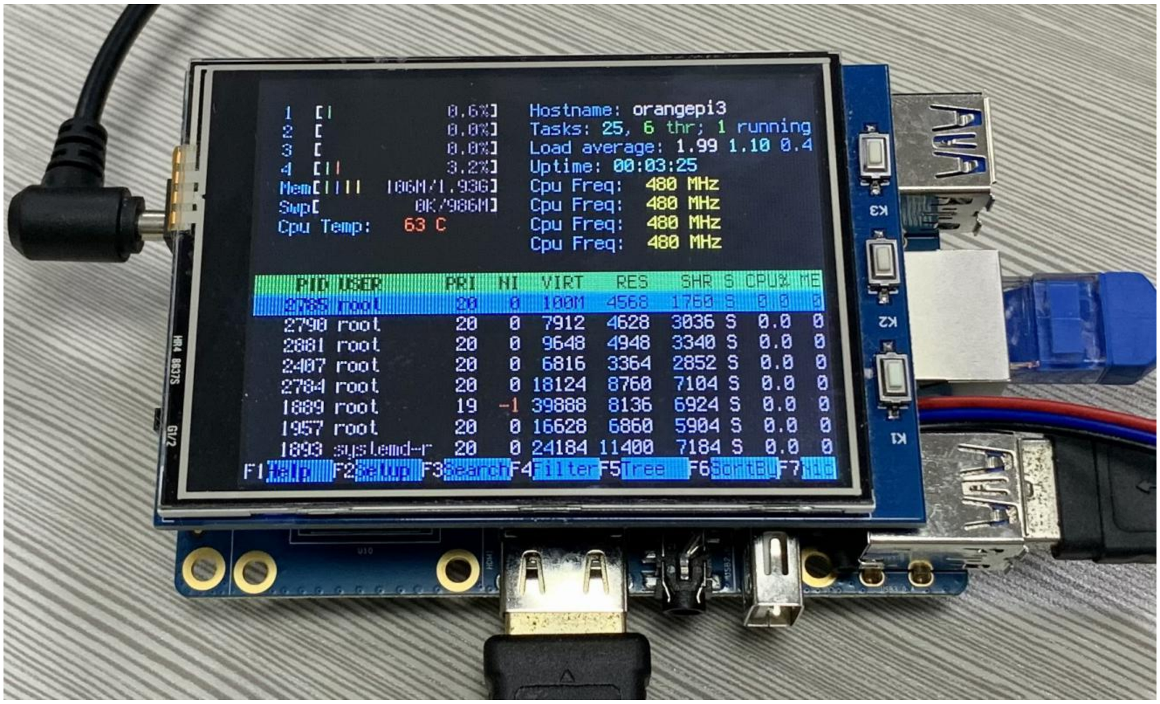
6) 还可以将 tty1 的输出映射到 LCD 屏幕的 fb 设备——**fb8**，映射完后，HDMI 就不会再有图像输出了

```
root@orangepi:~# con2fbmap 1 8
```

如果要切换回 HDMI 显示，请使用下面的命令

```
root@orangepi:~# con2fbmap 1 0
```

下面是运行 htop 命令的输出



7) 由于默认的终端字体太大，导致屏幕无法显示太多的内容，可以通过下面的方法来缩小终端的字体

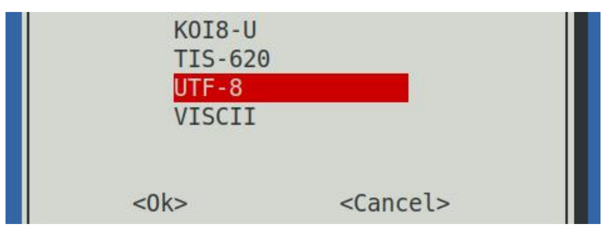
a. 首先运行 **dpkg-reconfigure console-setup**

root@orangepi:~# **apt-get update**

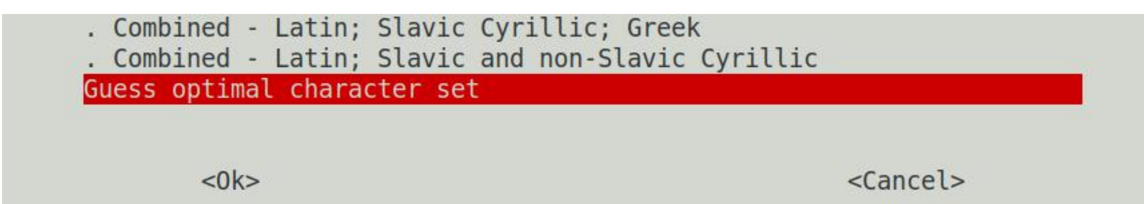
root@orangepi:~# **apt-get install kbd**

root@orangepi:~# **dpkg-reconfigure console-setup**

b. 终端编码选择 **UTF-8**



c. 然后选择 **Guess optimal character set**



d. 然后选择 **Terminus**

```
Fixed
Goha
GohaClassic
Terminus
TerminusBold
TerminusBoldVGA
VGA
Do not change the boot/kernel font
Let the system select a suitable font

<Ok>                                        <Cancel>
```

e. 最后选择字体大小为 6x12

```
6x12 (framebuffer only)
8x14
8x16
10x20 (framebuffer only)
11x22 (framebuffer only)
12x24 (framebuffer only)
14x28 (framebuffer only)
16x32 (framebuffer only)

<Ok>                                        <Cancel>
```

f. 设置完后就能看到 LCD 屏幕上的字体变小了

8) 设置系统启动自动加载 fbtft_device 模块的方法

a. 新建**/etc/modules-load.d/fbtft.con** 配置文件，文件内容如下所示

```
root@orangepi:~# cat /etc/modules-load.d/fbtft.conf
fbtft_device
```

b. 新建**/etc/modprobe.d/fbtft.conf** 配置文件，文件内容如下所示

```
root@orangepi:~# cat /etc/modprobe.d/fbtft.conf
options fbtft_device custom name=fb_ili9341  busnum=1 cs=0  gpios=reset:119,dc:362
rotate=90 speed=65000000 bgr=1 txbuflen=65536
```

c. 然后重启 linux 系统就能看到 fbtft_device 相关的内核模块都已自动加载

9) 如果希望 linux 系统启动后自动将 console 映射到 LCD 屏幕，请在 **/boot/orangepiEnv.txt** 中加入下面的配置，然后重启系统就能看到 LCD 屏幕有输出了

```
root@orangepi:~# cat /boot/orangepiEnv.txt | grep "fbcon"
```

```
extraargs=cma=25M fbcon=map:8
```

3.31.3. 3.5 寸 SPI LCD 显示屏

1) 测试的 LCD 显示屏详情页链接如下

```
http://www.lcdwiki.com/3.5inch_SPI_Module_ILI9488_SKU:MSP3520
```

2) LCD 显示屏和开发板的接线方式如下所示

TFT SPI 模块引脚	开发板 26pin 对应的引脚	GPIO -- GPIO num
VCC	1 号引脚	
GND	6 号引脚	
CS	24 号引脚	
RESET	12 号引脚	PD18 -- 114
DC/RS	16 号引脚	PD15 -- 111
SDI(MOSI)	19 号引脚	
SCK	23 号引脚	
LED	18 号引脚	PD16 -- 112
SDO(MISO)	21 引脚	

3) 将显示屏接到开发板后，再使用下面的命令加载 **fbtft_device** 内核模块

```
root@orangepi:~# modprobe fbtft_device custom name=fb_ili9488 busnum=1 cs=0 gpios=reset:114,dc:111,led:112 rotate=270 speed=65000000 bgr=1 txbuflen=65536
```

4) **fbtft_device** 内核模块加载时 dmesg 命令正确的输出 log 如下所示，而且由 log 可以知道 LCD 显示屏使用的 framebuffer 为 **fb8**

```
root@orangepi:~# dmesg | tail
[   36.897250] spidev spi1.0: dh2228fv spi1.0 1200kHz 8 bits mode=0x00
[   36.897709] spidev spi1.0: Deleting spi1.0
[   36.900560] fbtft_device: GPIOS used by 'fb_ili9488':
[   36.900579] fbtft_device: 'reset' = GPIO114
[   36.900591] fbtft_device: 'dc' = GPIO111
[   36.900602] fbtft_device: 'led' = GPIO112
[   36.900629] spi spi1.0: fb_ili9488 spi1.0 65000kHz 8 bits mode=0x00
[   36.918499] fb_ili9488: module is from the staging directory, the quality is unknown, you have been warned.
```

```
[    37.263271] graphics fb8: fb_ili9488 frame buffer, 480x320, 300 KiB video memory, 64 KiB buffer memory, fps=60, spi1.0 at 65 MHz
```

5) 然后使用下面的命令就可以在 LCD 显示屏上显示 Orange Pi 的 logo 图片

```
root@orangepi:~# apt update
root@orangepi:~# apt -y install fbi
root@orangepi:~# fbi -vt 1 -noverbose -d /dev/fb8 /boot/boot.bmp
```

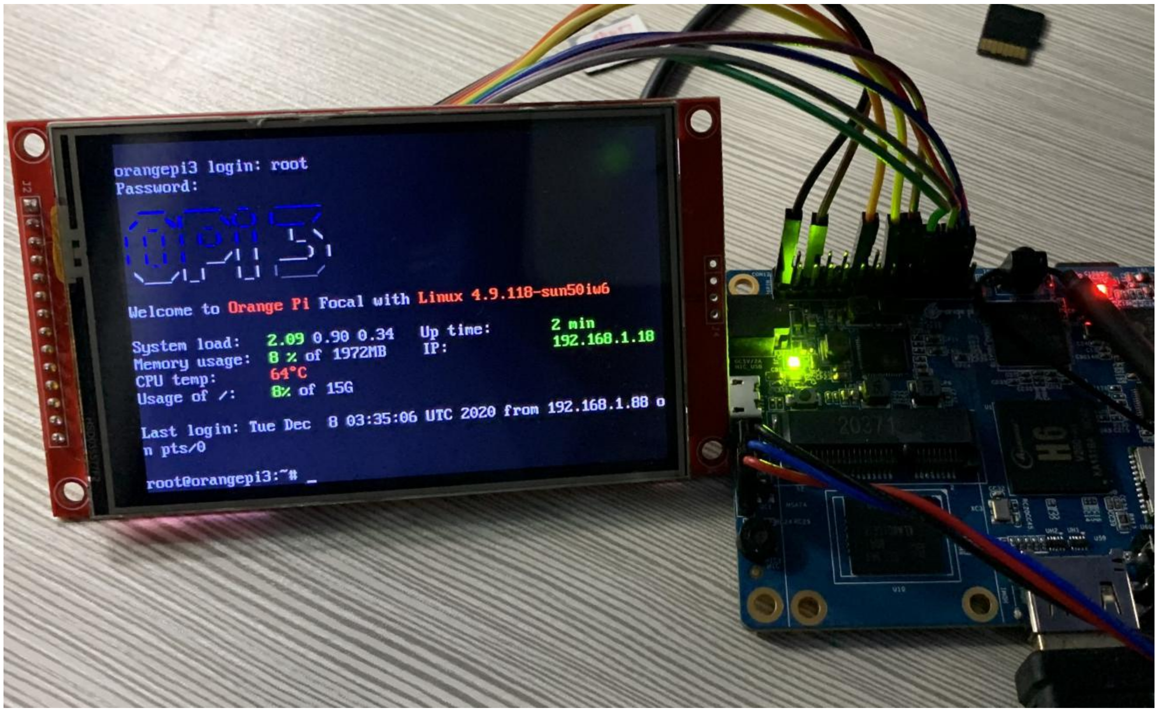


6) 还可以将 tty1 的输出映射到 LCD 显示屏的 fb 设备——**fb8**，映射完后，LCD 屏幕将会显示终端的输出，HDMI 就不会再有图像输出了

```
root@orangepi:~# con2fbmap 1 8
```

如果要切换回 HDMI 显示，请使用下面的命令

```
root@orangepi:~# con2fbmap 1 0
```



7) 设置系统启动自动加载 fbtft_device 模块的方法

a. 新建**/etc/modules-load.d/fbtft.con** 配置文件，文件内容如下所示

```
root@orangepi:~# cat /etc/modules-load.d/fbtft.conf
fbtft_device
```

b. 新建**/etc/modprobe.d/fbtft.conf** 配置文件，文件内容如下所示

```
root@orangepi:~# cat /etc/modprobe.d/fbtft.conf
options fbtft_device custom name=fb_ili9488  busnum=1 cs=0
gpios=reset:114,dc:111,led:112 rotate=270 speed=65000000 bgr=1 txbuflen=65536
```

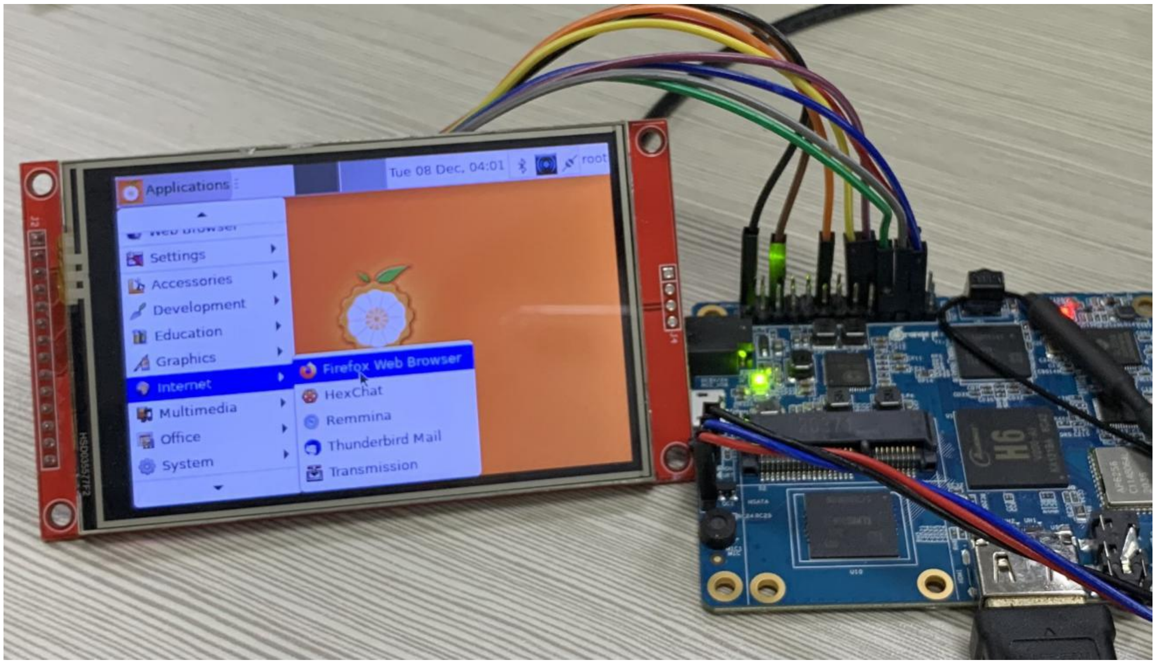
c. 然后重启 linux 系统就能看到 fbtft_device 相关的内核模块都已自动加载

8) 如果希望 linux 系统启动后自动将 console 映射到 LCD 显示屏，请在**/boot/orangepiEnv.txt** 中加入下面的配置，然后重启系统就能看到 LCD 显示屏有输出了

```
root@orangepi:~# cat /boot/orangepiEnv.txt | grep "fbcon"
extraargs=cma=25M fbcon=map:8
```

9) 如果需要将桌面版系统显示到 LCD 屏幕，可以执行下面的命令，等待几秒钟后，LCD 屏幕就能看到 linux 系统的桌面了

```
root@orangepi:~# FRAMEBUFFER=/dev/fb8 startx
```





10) 如果希望 linux 系统启动后自动将桌面显示到 LCD 显示屏，请在 linux 系统中添加下面的配置文件，然后重启系统就能看到 LCD 显示屏有显示输出了

```
root@orangepi:~# cat /usr/share/X11/xorg.conf.d/99-fbdev.conf
Section "Device"
   Identifier "myfb"
```

```
    Driver "fbdev"
    Option "fbdev" "/dev/fb8"
EndSection
```

3. 32. 将内核打印信息输出到 26pin 串口的方法

内核 console 默认输出到 ttyS0，也就是开发板上的 3pin 调试串口。我们也可以设置内核 console 输出重定向到 26pin 接口中的 UART3，方法如下：

1) linux5.4 系统需要先将 uart3 的配置打开，详细的配置方法见 26pin 的 UART 测试，linux4.9 系统默认就是打开的，无需额外的配置

2) 然后修改/boot/boot.cmd 中的 console=ttyS0 为 console=ttyS3

```
root@orangepi:~# vim /boot/boot.cmd
```

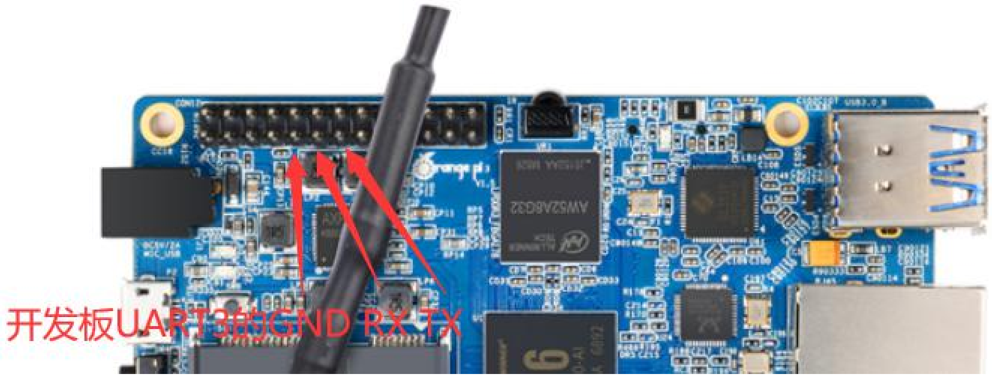
```
if test "${console}" = "display" || test "${console}" = "both"; then setenv consoleargs "console=ttyS3,115200 console=tty1"; fi
if test "${console}" = "serial"; then setenv consoleargs "console=ttyS3,115200"; fi
if test "${bootlogo}" = "true"; then setenv consoleargs "bootsplash.bootfile=bootsplash.orangepi ${consoleargs}"; fi
```

3) 然后将/boot/boot.cmd 重新编译为/boot/boot.scr（在开发板的 linux 系统中操作）

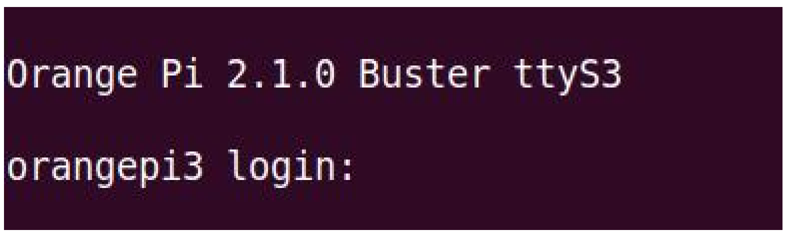
```
root@orangepi:~# mkimage -C none -A arm -T script -d /boot/boot.cmd /boot/boot.scr
Image Name:
Created:         Tue Dec   8 02:35:43 2020
Image Type:      ARM Linux Script (uncompressed)
Data Size:       2448 Bytes = 2.39 KiB = 0.00 MiB
Load Address: 00000000
Entry Point:     00000000
Contents:
   Image 0: 2440 Bytes = 2.38 KiB = 0.00 MiB
```

4) 然后将 USB 转 TTL 模块通过杜邦线接到 26pin 接口的 UART3 引脚上

a. USB 转 TTL 模块的 GND 接到开发板 26pin 接口的 GND 上
b. USB 转 TTL 模块的 **RX 接到开发板 UART3 的 TX 上**
c. USB 转 TTL 模块的 **TX 接到开发板 UART3 的 RX 上**



5) 然后重启开发板，可以看到内核 console 默认输出到了 ttyS3。注意此时 u-boot 的输出 log 还是输出到 ttyS0，不会输出到 ttyS3



3.33. 硬件看门狗测试

1) 下载 wiringOP 的代码

```
root@orangepi:~# apt update
root@orangepi:~# apt install git
root@orangepi:~# git clone https://github.com/orangepi-xunlong/wiringOP
```

2) 编译 watchdog 测试程序

```
root@orangepi:~# cd wiringOP/examples/
root@orangepi:~/wiringOP/examples# gcc watchdog.c -o watchdog
```

3) 运行看门狗测试程序

a. 第二个参数 10 表示看门狗的计数时间，如果这个时间内没有喂狗，系统会重启

b. 我们可以通过按下键盘上的任意键（ESC 除外）来喂狗，喂狗后，程序会打印一行 keep alive 表示喂狗成功

```
root@orangepi:~/wiringOP/examples# ./watchdog 10
open success
```

```
options is 33152,identity is sunxi-wdt
put_usr return,if 0,success:0
The old reset time is: 16
return ENOTTY,if -1,success:0
return ENOTTY,if -1,success:0
put_user return,if 0,success:0
put_usr return,if 0,success:0
keep alive
keep alive
keep alive
```

3.34. 查看 H6 芯片的 chipid

注意：此方法只适用于 linux4.9 系统

1) 查看 h6 芯片 chipid 的命令如下所示，每个芯片的 chipid 都是不同的，所以可以使用 chipid 来区分多个开发板

```
root@orangepi:~# cat /sys/class/sunxi_info/sys_info | grep "chipid"
sunxi_chipid          : 38641f0f0141410900004c0000000000
```

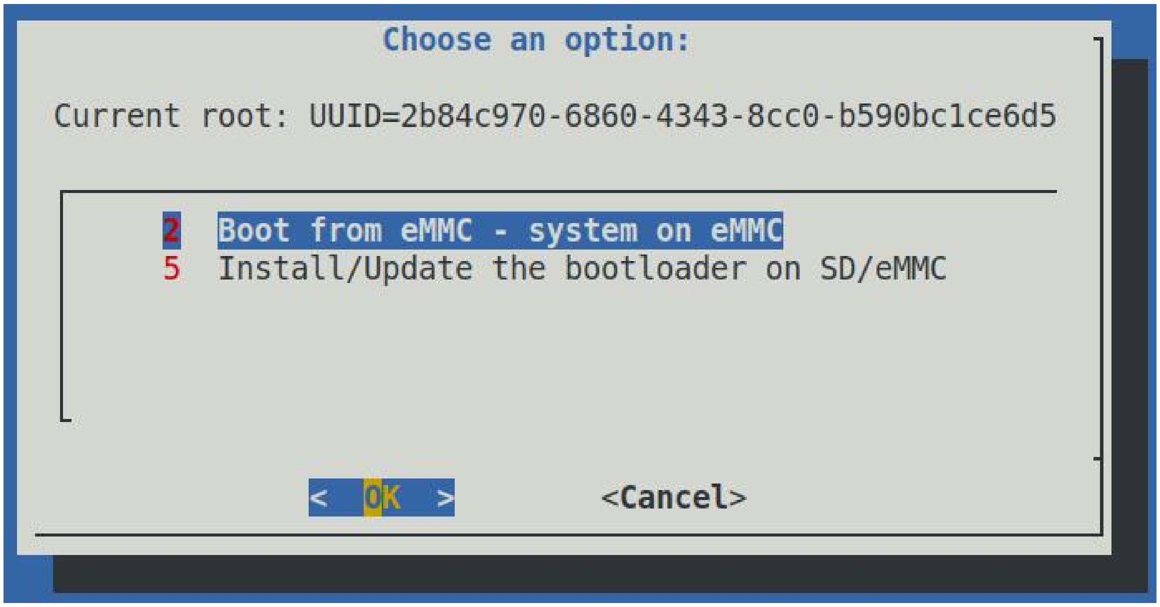
3.35. 烧写 linux 镜像到 eMMC 的方法

1) 烧录 linux 镜像到 eMMC 中需要借助 TF 卡来完成，首先将 linux 镜像烧录到 TF 卡上，然后启动开发板进入 linux 系统

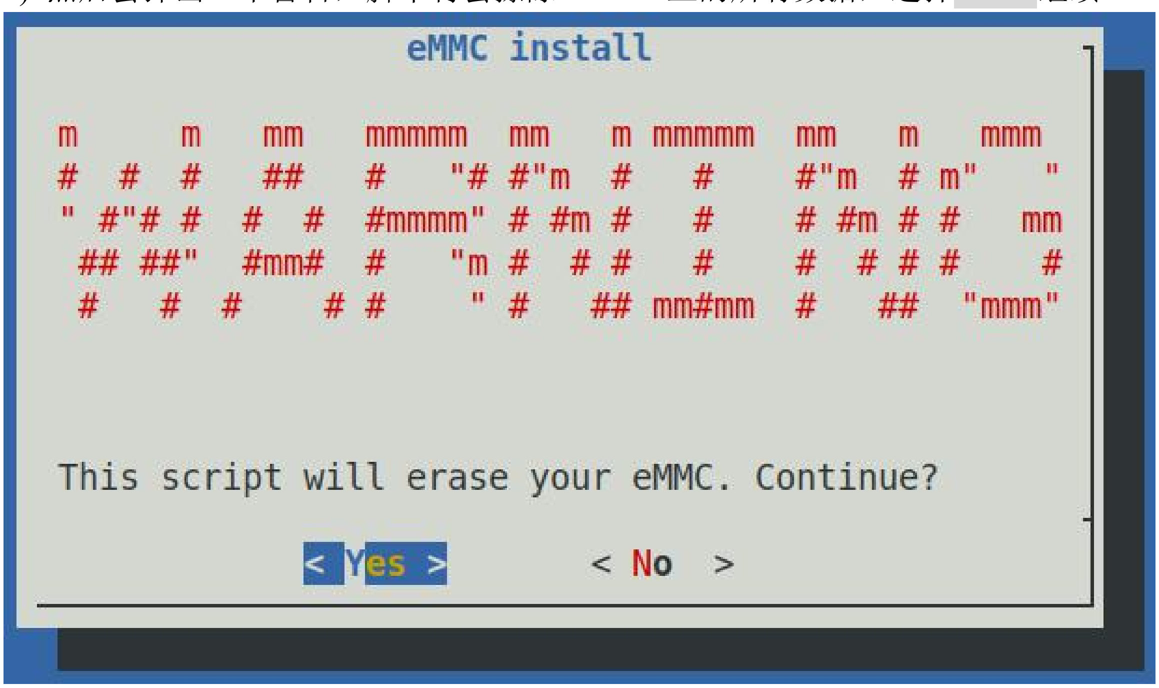
2) 然后运行 **nand-sata-install** 脚本

```
root@orangepi:~# nand-sata-install
```

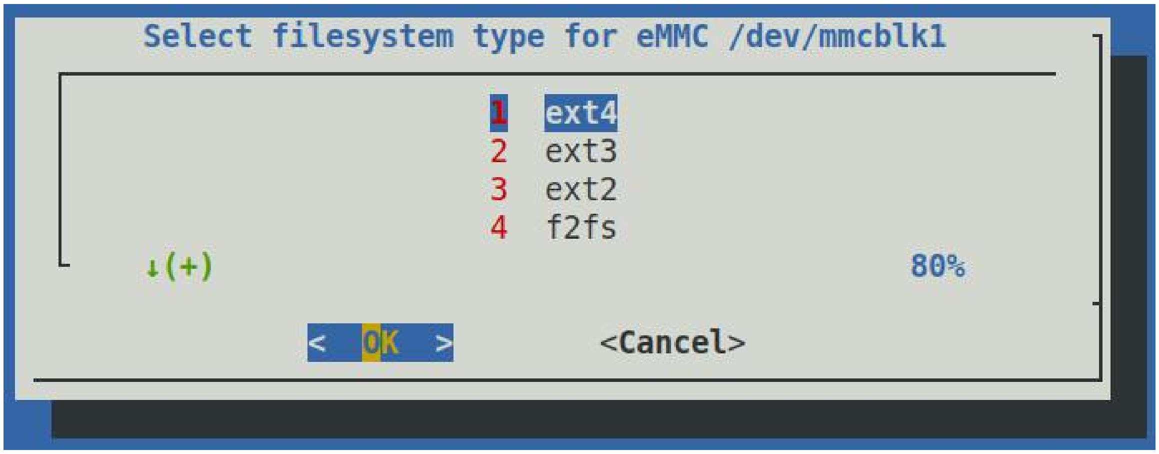
3) 然后选择 **2 Boot from eMMC - sysytem on eMMC**

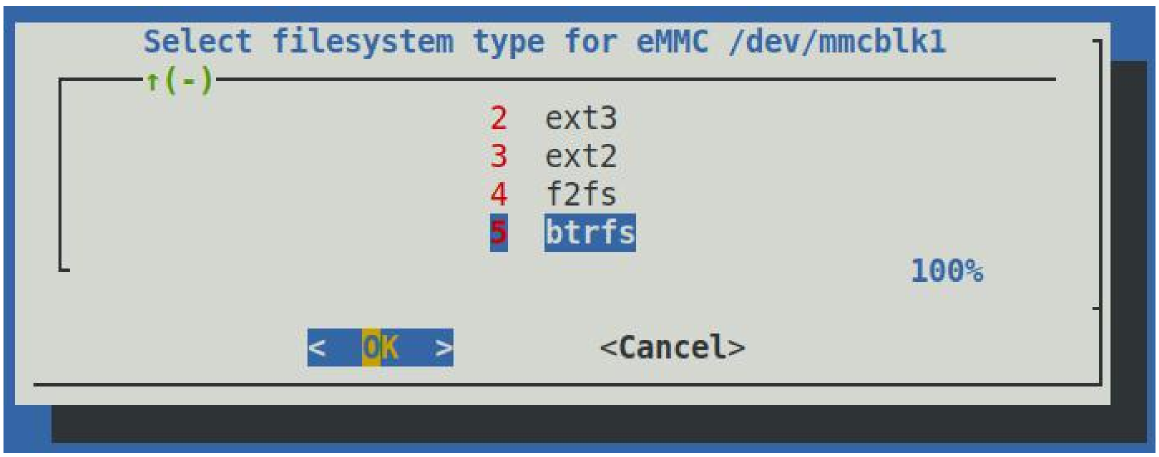


4) 然后会弹出一个警告，脚本将会擦除 eMMC 上的所有数据，选择**<Yes>**继续

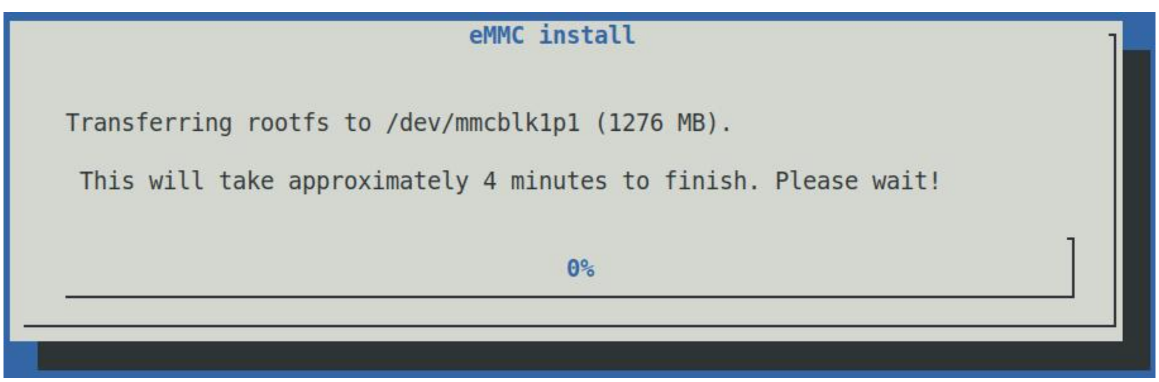


5) 然后会提示选择文件系统的类型，支持 ext2/3/4、f2fs 以及 btrfs 五种文件系统

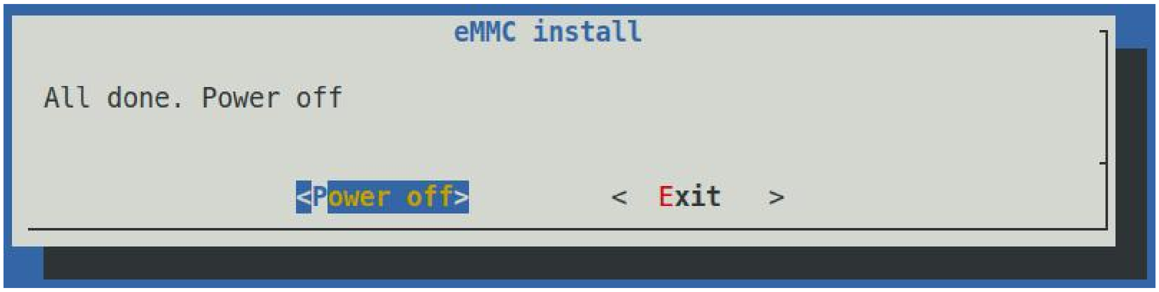




6) 然后会开始格式化 eMMC，格式化完 eMMC 后就会开始烧录 linux 镜像到 eMMC 中



7) 烧录完后会提示下面的选项，可以选择**<Power off>**直接关机



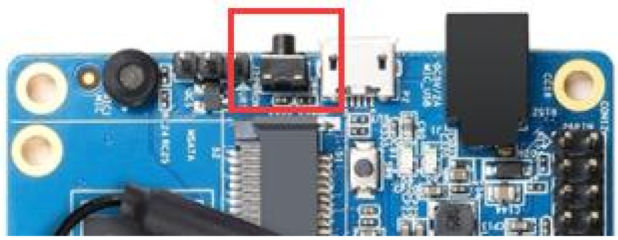
8) 然后拔出 TF 卡，再重新上电，就会启动 eMMC 中的 linux 系统了

3.36. 开机和关机方法

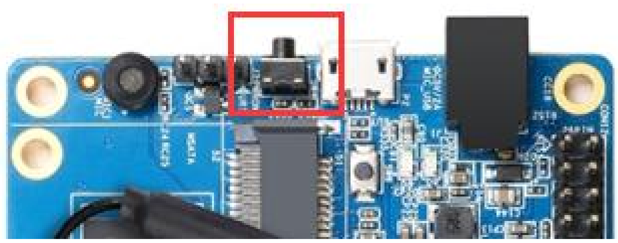
1) 使用 poweroff 命令关机

```
root@orangepi:~# poweroff
```

2) 还可以短按开发板上的电源按键来关机



3) 关机后长按开发板上的电源按键 **2** 秒即可开机



4) 重启 linux 系统的命令为

```
root@orangepi:~# reboot
```

4. Linux SDK 使用说明

Linux SDK 的编译都是在安装有 **Ubuntu 18.04 的 PC 或者虚拟机（VirtualBox 或 VMware）**上进行的，请不要使用其他版本的 Ubuntu 系统或者在 WSL 上编译 Linux SDK

4. 1. 获取 linux sdk 的源码

4. 1. 1. 从 github 下载 orangepi-build

1) 首先下载 orangepi-build 的代码，orangepi-build 的代码是基于 armbian build 编译系统修改而来的，目前 H6 系列开发板已经支持 legacy 分支和 current 分支

```
test@test:~$ sudo apt update
test@test:~$ sudo apt install git
test@test:~$ git clone https://github.com/orangepi-xunlong/orangepi-build.git
```

2) legacy 分支一般使用 BSP 版本的 u-boot 和内核，current 分支一般使用接近主线版本的 u-boot 和内核，H6 系列开发板当前使用的 u-boot 和 linux 内核如下所示

分支	u-boot 版本	linux 内核版本
legacy	u-boot 2014.07	linux4.9.118
current	u-boot 2020.04	linux5.4.65

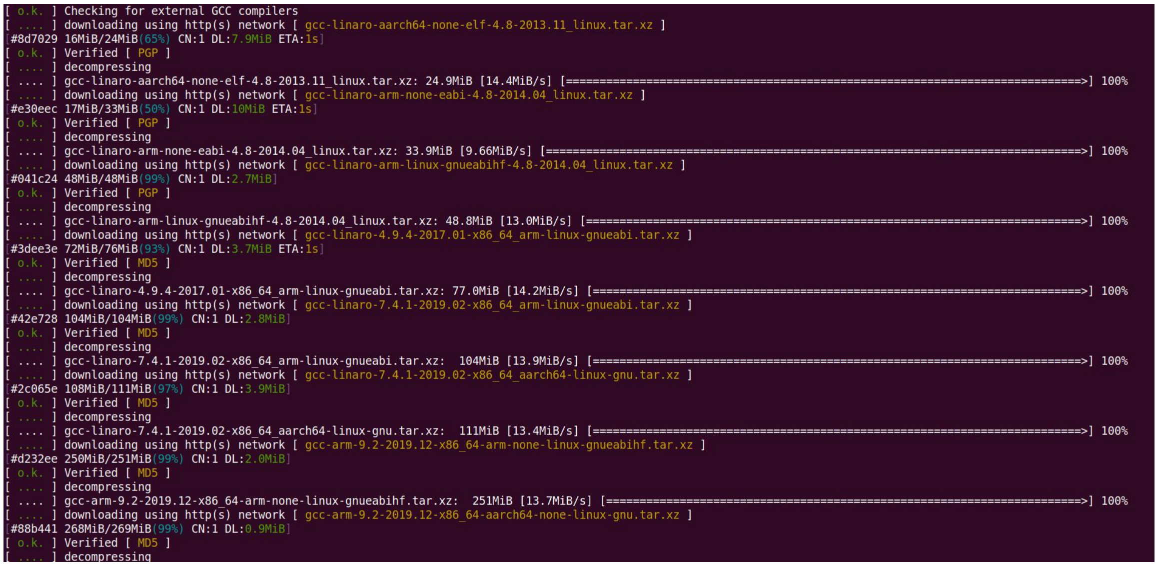
3) orangepi-build 下载完后会包含下面的文件和文件夹

a. **build.sh**: 编译启动脚本
b. **external**: 包含编译镜像需要用的配置文件、特定的脚本以及部分程序的源码等
c. **LICENSE**: GPL 2 许可证文件
d. **README.md**: orangepi-build 说明文件
e. **scripts**: 编译 linux 镜像的通用脚本

```
test@test:~/orangepi-build$ ls
build.sh    external    LICENSE    README.md    scripts
```

4.1.2. 下载交叉编译工具链

1) orangepi-build 第一次运行的时候会自动下载交叉编译工具链放在 **toolchains** 文件夹中，每次运行 orangepi-build 的 build.sh 脚本后，都会检查 **toolchains** 中的交叉编译工具链是否都存在，如果不存在则会重新开始下载，如果存在则直接使用，不会重复下载



```
[ o.k. ] Checking for external GCC compilers
[ .... ] downloading using http(s) network [ gcc-linaro-aarch64-none-elf-4.8-2013.11_linux.tar.xz ]
[#8d7029 16MiB/24MiB(65%) CN:1 DL:7.9MiB ETA:1s]
[ o.k. ] Verified [ PGP ]
[ .... ] decompressing
[ .... ] gcc-linaro-aarch64-none-elf-4.8-2013.11_linux.tar.xz: 24.9MiB [14.4MiB/s] [==========================================================================>] 100%
[ .... ] downloading using http(s) network [ gcc-linaro-arm-none-eabi-4.8-2014.04_linux.tar.xz ]
[#e30eec 17MiB/33MiB(50%) CN:1 DL:10MiB ETA:1s]
[ o.k. ] Verified [ PGP ]
[ .... ] decompressing
[ .... ] gcc-linaro-arm-none-eabi-4.8-2014.04_linux.tar.xz: 33.9MiB [9.66MiB/s] [==========================================================================>] 100%
[ .... ] downloading using http(s) network [ gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux.tar.xz ]
[#041c24 48MiB/48MiB(99%) CN:1 DL:2.7MiB]
[ o.k. ] Verified [ PGP ]
[ .... ] decompressing
[ .... ] gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux.tar.xz: 48.8MiB [13.0MiB/s] [==========================================================================>] 100%
[ .... ] downloading using http(s) network [ gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi.tar.xz ]
[#3dee3e 72MiB/76MiB(93%) CN:1 DL:3.7MiB ETA:1s]
[ o.k. ] Verified [ MD5 ]
[ .... ] decompressing
[ .... ] gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi.tar.xz: 77.0MiB [14.2MiB/s] [==========================================================================>] 100%
[ .... ] downloading using http(s) network [ gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi.tar.xz ]
[#42e728 104MiB/104MiB(99%) CN:1 DL:2.8MiB]
[ o.k. ] Verified [ MD5 ]
[ .... ] decompressing
[ .... ] gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi.tar.xz:  104MiB [13.9MiB/s] [==========================================================================>] 100%
[ .... ] downloading using http(s) network [ gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu.tar.xz ]
[#2c065e 108MiB/111MiB(97%) CN:1 DL:3.9MiB]
[ o.k. ] Verified [ MD5 ]
[ .... ] decompressing
[ .... ] gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu.tar.xz:  111MiB [13.4MiB/s] [==========================================================================>] 100%
[ .... ] downloading using http(s) network [ gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabihf.tar.xz ]
[#d232ee 250MiB/251MiB(99%) CN:1 DL:2.0MiB]
[ o.k. ] Verified [ MD5 ]
[ .... ] decompressing
[ .... ] gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabihf.tar.xz:  251MiB [13.7MiB/s] [==========================================================================>] 100%
[ .... ] downloading using http(s) network [ gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu.tar.xz ]
[#88b441 268MiB/269MiB(99%) CN:1 DL:0.9MiB]
[ o.k. ] Verified [ MD5 ]
[ .... ] decompressing
```

2) 交叉编译工具链在中国境内的镜像网址为清华大学的开源软件镜像站

https://mirrors.tuna.tsinghua.edu.cn/armbian-releases/_toolchain/

3) **toolchains** 下载完后会包含多个版本的交叉编译工具链

```
test@test:~/orangepi-build$ ls toolchains/
gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu
gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi
gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabihf
gcc-linaro-aarch64-none-elf-4.8-2013.11_linux
gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi
gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux
gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu
gcc-linaro-arm-none-eabi-4.8-2014.04_linux
```

4) 编译 H6 linux 内核源码使用的交叉编译工具链为

a. linux4.9

```
gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu
```

b. linux5.4

```
gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu
```

5) 编译 H6 u-boot 源码使用的交叉编译工具链为

a. u-boot 2014.07

```
gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi
```

b. u-boot 2020.04

```
gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu
```

4.1.3. orangepi-build 完整目录结构说明

1) orangepi-build 仓库下载完后并不包含 linux 内核、u-boot 的源码以及交叉编译工具链，linux 内核和 u-boot 的源码存放在独立的 git 仓库中（**请不要单独下载使用内核和 u-boot 的源码进行编译操作，除非你知道怎么用**）

a. linux 内核源码存放的 git 仓库如下，其中 sun50iw6 为 H6 SOC 芯片代号

a) linux4.9

```
https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-4.9-sun50iw6
```

b) linux5.4

```
https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-5.4
```

b. u-boot 源码存放的 git 仓库如下，其中 sun50iw6 为 H6 SOC 芯片代号

a) u-boot 2014.07

```
https://github.com/orangepi-xunlong/u-boot-orangepi/tree/v2014.07-sun50iw6-linux4.9
```

b) u-boot 2020.04

```
https://github.com/orangepi-xunlong/u-boot-orangepi/tree/v2020.04
```

2) orangepi-build 第一次运行的时候会去下载交叉编译工具链、u-boot 和 linux 内核源码，成功编译完一次 linux 镜像后在 orangepi-build 中可以看到的文件和文件夹有

a. **build.sh**: 编译启动脚本

b. **external**: 包含编译镜像需要用的配置文件、特定功能的脚本以及部分程序的源码，编译镜像过程中缓存的 rootfs 压缩包也存放在 external 中

c. **kernel**: 存放 linux 内核的源码，里面名为 **orange-pi-4.9-sun50iw6** 的文件夹存放的就是 H6 开发板 legacy 分支的内核源码，里面名为 **orange-pi-5.4** 的文件夹存放的就是 H6 开发板 current 分支的内核源码（如果只编译了 legacy 分支的 linux 镜像，那么则只能看到 legacy 分支的内核源码；如果只编译了 current 分支的 linux 镜像，那么则只能看到 current 分支的内核源码），内核

源码的文件夹的名字请不要手动修改，如果修改了编译系统运行时会重新下载内核源码

d. **LICENSE**: GPL 2 许可证文件
e. **README.md**: orangepi-build 说明文件
f. **output**: 存放编译生成的 u-boot、linux 等 deb 包、编译日志以及编译生成的镜像等文件
g. **scripts**: 编译 linux 镜像的通用脚本
h. **toolchains**: 存放交叉编译工具链
i. **u-boot**: 存放 u-boot 的源码，里面名为 **v2014.07-sun50iw6-linux4.9** 的文件夹存放的就是 H6 开发板 legacy 分支的 u-boot 源码，里面名为 **v2020.04** 的文件夹存放的就是 H6 开发板 current 分支的 u-boot 源码(如果只编译了 legacy 分支的 linux 镜像，那么则只能看到 legacy 分支的 u-boot 源码；如果只编译了 current 分支的 linux 镜像，那么则只能看到 current 分支的 u-boot 源码)，u-boot 源码的文件夹的名字请不要手动修改，如果修改了编译系统运行时会重新下载 u-boot 源码
j. **userpatches**: 存放编译脚本需要用到的配置文件

```
test@test:~/orangepi-build$ ls
build.sh   external   kernel   LICENSE   output   README.md   scripts   toolchains
u-boot   userpatches
```

4.1.4. 从百度云盘下载 orangepi-build

1) 如果从 github 下载 orangepi-build 的速度很慢，还可以从百度云盘下载 orangepi-build 的压缩包，百度云盘的下载链接为

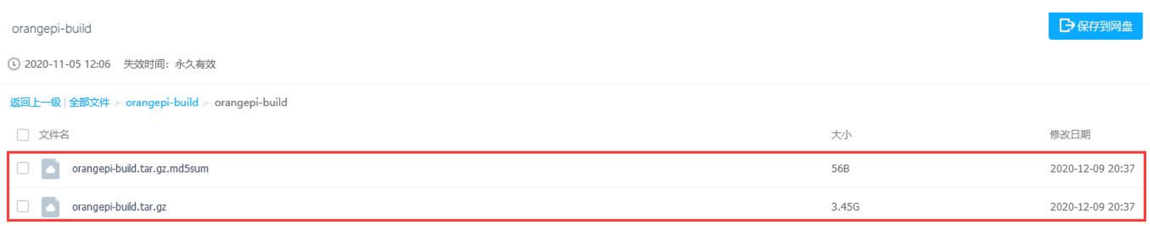
```
链接：https://pan.baidu.com/s/1vWQmCmSYdH7iCDFyKpJtVw
提取码：zero
```



2) 百度云盘的 orangepi-build 文件夹下有两个文件

a. **orangepi-build.tar.gz** 为 orangepi-build 源码的压缩包
b. **orangepi-build.tar.gz.md5sum** 为 orangepi-build 源码的压缩包的 MD5 校验和文件
c. 下载完后，请首先检查下 orangepi-build.tar.gz 压缩包的 MD5 校验和是否正确，这样可以防止下载的压缩包有问题，如果不正确，请重新下载，检查校验和是否正确的命令为

```
test@test:~$ md5sum -c orangepi-build.tar.gz.md5sum
orangepi-build.tar.gz: 成功
```



3) 然后就可以使用 tar -zxf 命令解压 orangepi-build.tar.gz

```
test@test:~$ tar -zxf orangepi-build.tar.gz
test@test:~$ cd orangepi-build/
test@test:~/orangepi-build$ ls
build.sh    external    kernel    LICENSE    README.md    scripts    toolchains
u-boot    userpatches
```

4) 使用 orangepi-build 编译系统前，请先将 orangepi-build 和 github 服务器进行同步，确保代码为最新的状态

```
test@test:~/orangepi-build$ git pull
```

如果 orangepi-build 和 github 服务器同步过程有问题，可以尝试下面的方法

```
test@test:~/orangepi-build$ sudo rm -rf external scripts
test@test:~/orangepi-build$ git checkout .
test@test:~/orangepi-build$ git pull
```

5) 百度云盘上的 **orangepi-build.tar.gz** 压缩包除了包含 orangepi-build 编译系统的代码外，还缓存了交叉编译工具链，u-boot 和 linux 内核的源码。所以在编译镜像的过程中就不会去 github 服务器上从头开始下载 u-boot 和 linux 内核的源码以及交叉编译工具链，这样可以节省大量的时间。orangepi-build 编译系统开始运行时，默认会自动从 github 同步 u-boot 和 linux 内核的源码，以确保代码为最新状态，所以无需

手动同步 u-boot 和 linux 内核的源码

6) 百度云盘上的 **orangepi-build.tar.gz** 解压后的 kernel 文件夹中会包含多个版本的内核源码，请不要修改这些内核源码文件夹的名字，否则会导致编译系统找不到内核源码进而重新从 github 下载内核源码。kernel 文件夹下的多个内核源码并非都能被 H6 系列开发板使用，H6 系列开发板使用到的内核源码有：

orange-pi-4.9-sun50iw6	H6 系列开发板 legacy 分支使用
orange-pi-5.4	H6 系列开发板 current 分支使用
~~orange-pi-4.9-sun50iw9~~	~~H616 开发板使用，H6 系列开发板可以忽略~~

7) 百度云盘上的 **orangepi-build.tar.gz** 解压后的 u-boot 文件夹中会包含多个版本的 u-boot 源码，请不要修改这些 u-boot 源码文件夹的名字，否则会导致编译系统找不到 u-boot 源码进而重新从 github 下载 u-boot 源码。u-boot 文件夹下的多个版本的 u-boot 源码并非都能被 H6 系列开发板使用，H6 系列开发板使用到的 u-boot 源码有：

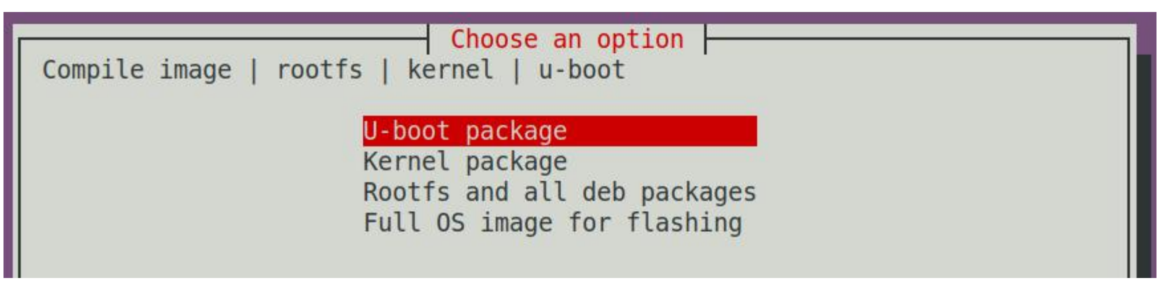
v2014.07-sun50iw6-linux4.9	H6 系列开发板 legacy 分支使用
v2020.04	H6 系列开发板 current 分支使用
~~v2018.05-sun50iw9~~	~~H616 开发板使用，H6 系列开发板可以忽略~~

4.2. 编译 u-boot

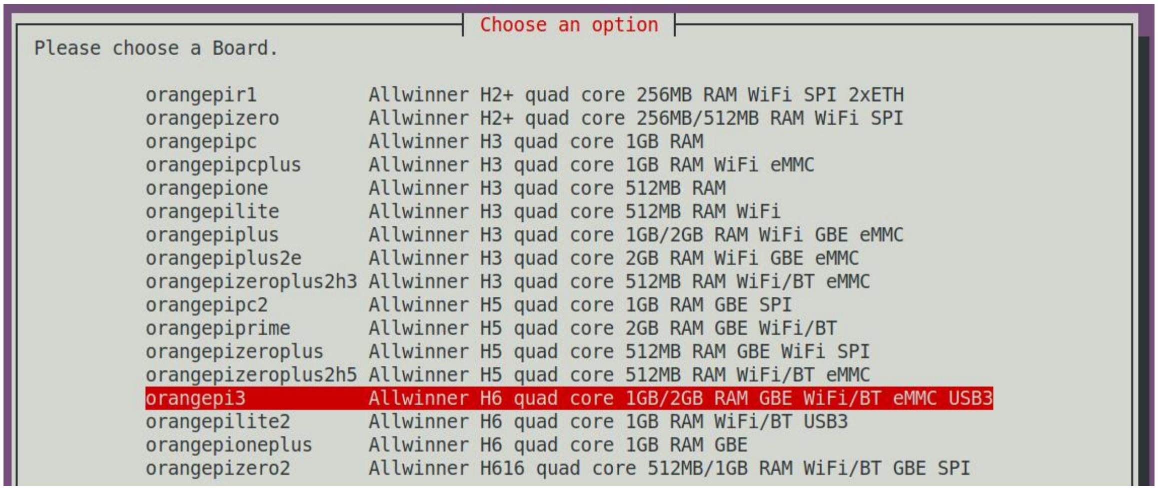
1) 运行 build.sh 脚本，记得加 sudo 权限

```
test@test:~/orangepi-build$ sudo ./build.sh
```

2) 选择 **U-boot package**，然后回车



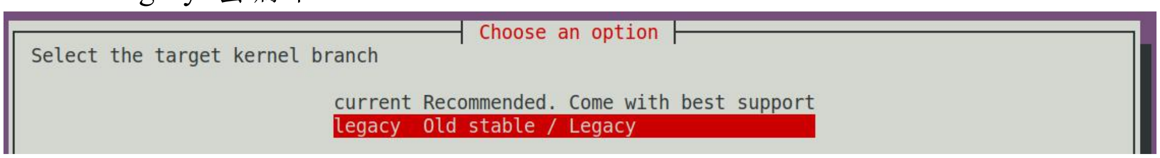
3) 接着选择开发板的型号



4) 然后选择分支

a. current 会编译 u-boot v2020.04

b. legacy 会编译 u-boot v2014.07



5) 然后就会开始编译 u-boot，编译时提示的部分信息说明如下（以 legacy 分支为例）

a. u-boot 源码的版本

[o.k.] Compiling u-boot [**v2014.07**]

b. 交叉编译工具链的版本

[o.k.] Compiler version [**arm-linux-gnueabi-gcc 4.9.4**]

c. 编译生成的 u-boot deb 包的路径

[o.k.] Target directory [**output/debs/u-boot**]

d. 编译生成的 u-boot deb 包的包名

[o.k.] File name [**linux-u-boot-legacy-orangepi3_2.1.0_arm64.deb**]

e. 编译使用的时间

[o.k.] Runtime [**1 min**]

f. 重复编译 u-boot 的命令，使用下面的命令无需通过图形界面选择，可以直接开始编译 u-boot

[o.k.] Repeat Build Options [**sudo ./build.sh BOARD=orangepi3 BRANCH=legacy BUILD_OPT=u-boot KERNEL_CONFIGURE=no**]

6) 查看编译生成的 u-boot deb 包

```
test@test:~/orangepi-build$ ls output/debs/u-boot/
linux-u-boot-legacy-orangepi3_2.1.0_arm64.deb
```

7) 生成的 u-boot 的 deb 包包含的文件如下所示

a. 使用下面的命令可以解压 deb 包

```
test@test:~/orangepi-build$ cd output/debs/u-boot
test@test:~/orangepi_build/output/debs/u-boot$ $ dpkg -x \
linux-u-boot-legacy-orangepi3_2.1.0_arm64.deb .    (注意命令最后有个“.”)
test@test:~/orangepi_build/output/debs/u-boot$ ls
linux-u-boot-current-orangepi3_2.1.0_armhf.deb    usr
```

b. 解压后的文件如下所示

```
test@test:~orangepi-build/output/debs/u-boot$ tree usr
usr
└── lib
    ├── linux-u-boot-legacy-orangepi3_2.1.0_arm64
    │   ├── boot0_sdcard.fex          //boot0 的二进制文件
    │   └── boot_package.fex          //u-boot 的二进制文件
    └── u-boot
        ├── LICENSE
        ├── orangepi3-u-boot.dts
        └── platform_install.sh


3 directories, 5 files
```

8) orangepi-bulid 编译系统编译 u-boot 源码时首先会将 u-boot 的源码和 github 服务器的 u-boot 源码进行同步，所以如果想修改 u-boot 的源码，首先需要关闭源码的下载更新功能（**需要完整编译过一次 u-boot 后才能关闭这个功能，否则会提示找不到 u-boot 的源码**），否则所作的修改都会被还原，方法如下：

设置 userpatches/config-default.conf 中的 IGNORE_UPDATES 变量为“yes”

```
test@test:~/orangepi-build$ vim userpatches/config-default.conf
IGNORE_UPDATES="yes"
```

9) 调试 u-boot 代码时，可以使用下面的方法来更新 linux 镜像中的 u-boot 进行测试

a. 将编译好的 u-boot 的 deb 包上传到开发板的 linux 系统中

```
test@test:~/orangepi-build$ cd output/debs/u-boot
test@test:~/orangepi_build/output/debs/u-boot$ scp \
linux-u-boot-legacy-orangepi3_2.1.0_arm64.deb root@192.168.1.xxx:/root
```

b. 然后登录到开发板，卸载已安装的 u-boot 的 deb 包

```
root@orangepi:~# apt purge -y linux-u-boot-orangepi3-legacy
```

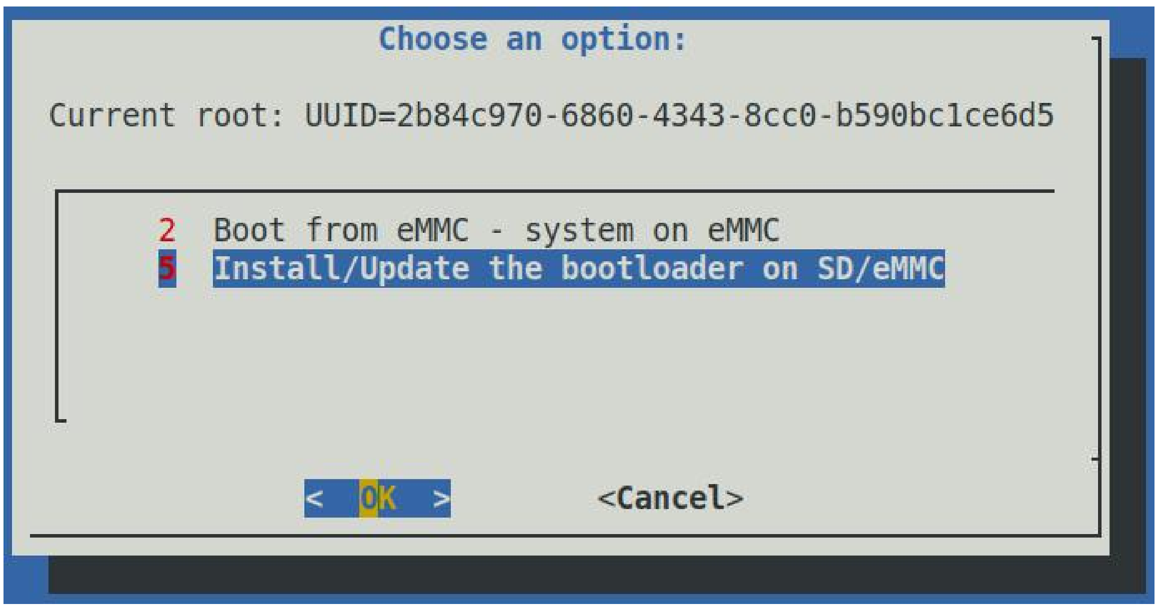
c. 再安装刚才上传的新的 u-boot 的 deb 包

```
root@orangepi:~# dpkg -i linux-u-boot-legacy-orangepi3_2.1.0_arm64.deb
```

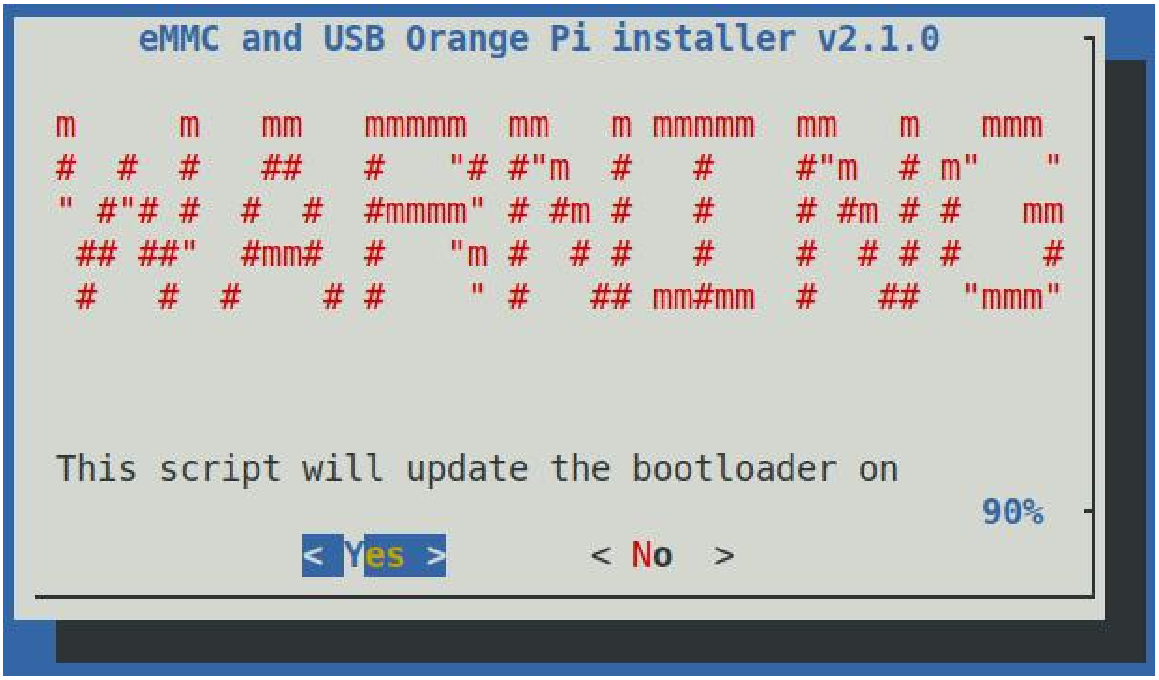
d. 然后运行 nand-sata-install 脚本

```
root@orangepi:~# nand-sata-install
```

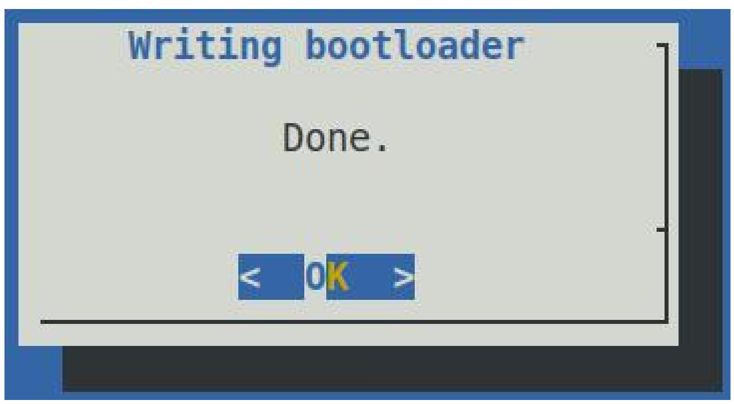
e. 然后选择 **5 Install/Update the bootloader on SD/eMMC**



f. 按下回车键后首先会弹出一个 Warring



g. 再按下回车键就会开始更新 u-boot，更新完后会显示下面的信息



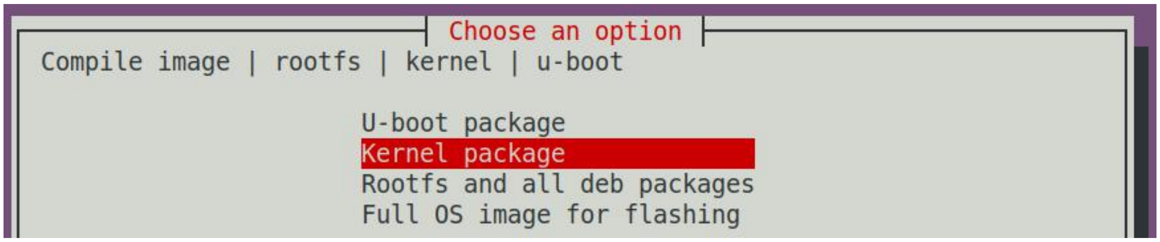
h. 然后就可以重启开发板来测试 u-boot 的修改是否生效了

4.3. 编译 linux 内核

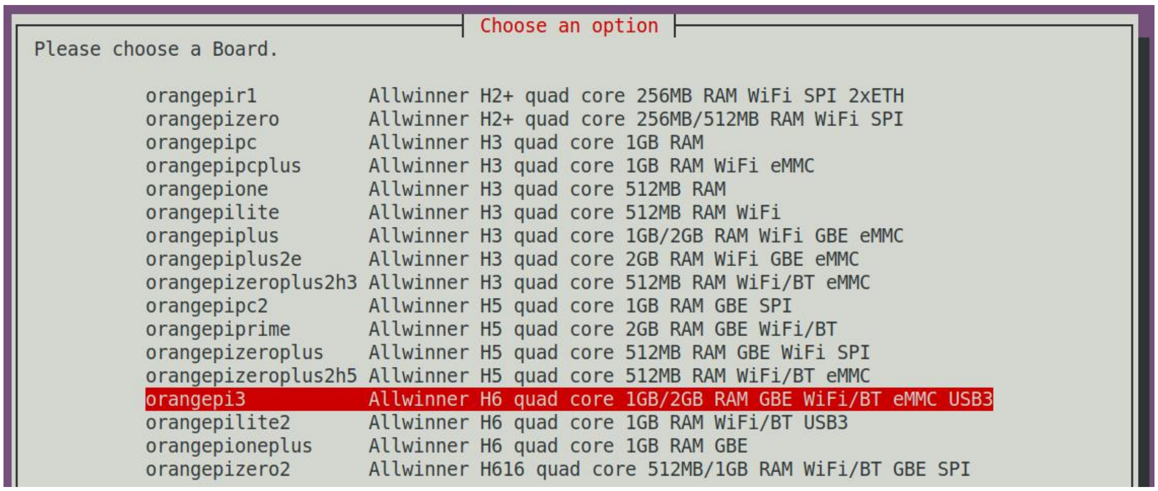
1) 运行 build.sh 脚本，记得加 sudo 权限

```
test@test:~/orangepi-build$ sudo ./build.sh
```

2) 选择 **Kernel package**，然后回车

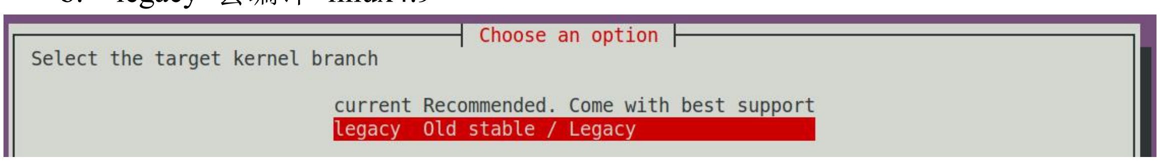


3) 接着选择开发板的型号

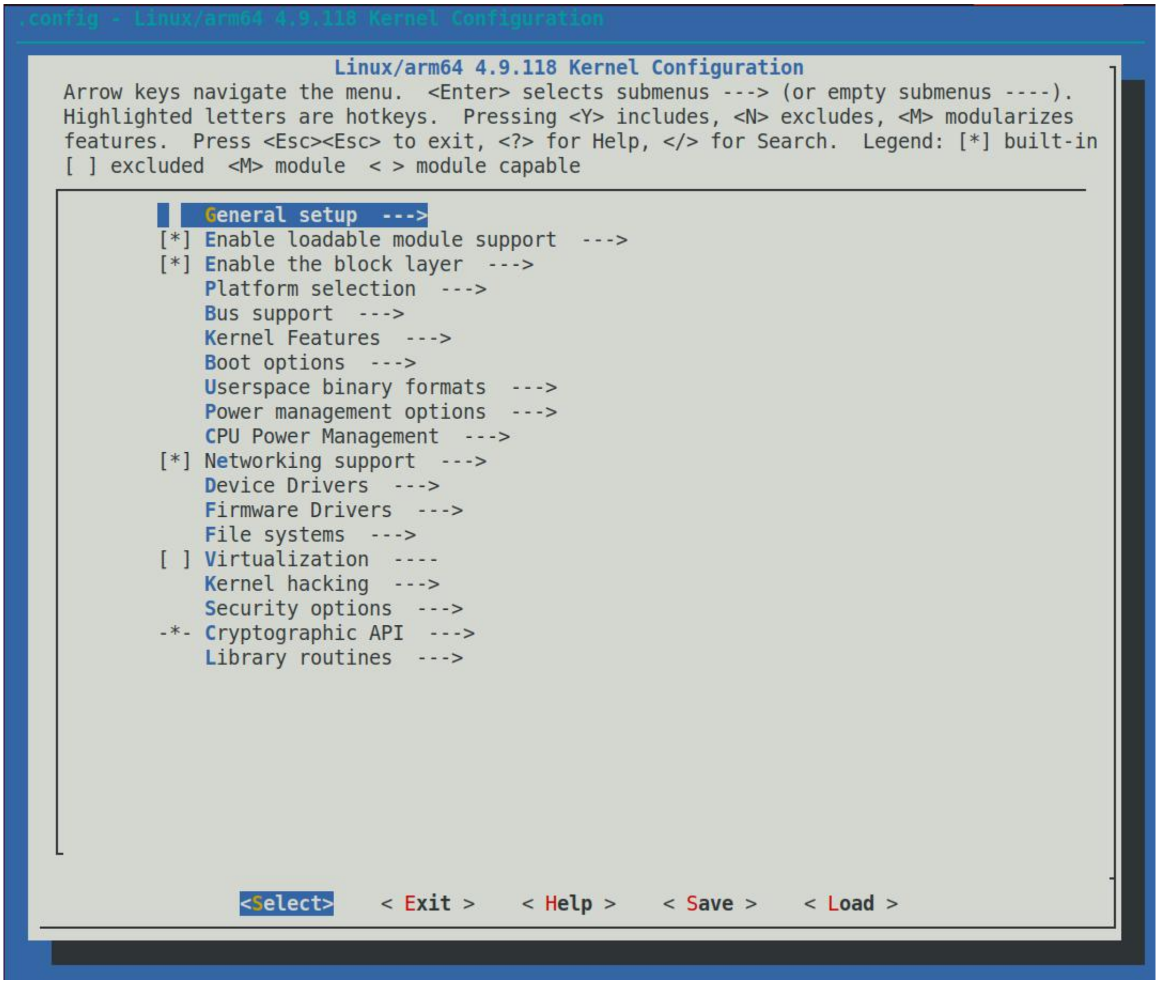


4) 然后选择分支

a. current 会编译 linux 5.4
b. legacy 会编译 linux4.9



5) 然后会弹出通过 **make menuconfig** 打开的内核配置的界面，此时可以直接修改内核的配置，如果不需要修改内核配置，直接退出即可，退出后会开始编译内核源码



a. 如果不需要修改内核的配置选项，在运行 build.sh 脚本时，传入 **KERNEL_CONFIGURE=no** 就可临时屏蔽弹出内核的配置界面了

```
test@test:~/orangepi-build$ sudo ./build.sh KERNEL_CONFIGURE=no
```

b. 也可以设置 orangepi-build/userpatches/config-default.conf 配置文件中的 **KERNEL_CONFIGURE=no**，这样可以永久禁用这个功能

c. 编译内核的时候如果提示下面的错误，这是由于 Ubuntu PC 的终端界面太小，导致 make menuconfig 的界面无法显示，请把 Ubuntu PC 的终端调到最大，然后重新运行 build.sh 脚本

```
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTLD  scripts/kconfig/mconf
scripts/kconfig/mconf  Kconfig
Your display is too small to run Menuconfig!
It must be at least 19 lines by 80 columns.
scripts/kconfig/Makefile:28: recipe for target 'menuconfig' failed
make[1]: *** [menuconfig] Error 1
Makefile:560: recipe for target 'menuconfig' failed
make: *** [menuconfig] Error 2
[ error ] ERROR in function compile_kernel [ compilation.sh:376 ]
[ error ] Error kernel menuconfig failed
[ o.k. ] Process terminated
```

6) 编译内核源码时提示的部分信息说明如下（以 legacy 分支为例）

a. linux 内核源码的版本

[o.k.] Compiling legacy kernel [**4.9.118**]

b. 使用的交叉编译工具链的版本

[o.k.] Compiler version [**aarch64-none-linux-gnu-gcc 9.2.1**]

c. 内核默认使用的配置文件以及它存放的路径

[o.k.] Using kernel config file [**config/kernel/linux-sun50iw6-legacy.config**]

d. 如果设置了 **KERNEL_CONFIGURE=yes**，内核最终使用的配置文件.config 会复制到 **output/config** 中，如果没有对内核配置进行修改，最终的配置文件和默认的配置文件是一致的

[o.k.] Exporting new kernel config [**output/config/linux-sun50iw6-legacy.config**]

e. 编译生成的内核相关的 deb 包的路径

[o.k.] Target directory [**output/debs/**]

f. 编译生成的内核镜像 deb 包的包名

[o.k.] File name [**linux-image-legacy-sun50iw6_2.1.0_arm64.deb**]

g. 编译使用的时间

[o.k.] Runtime [**5 min**]

h. 最后会显示重复编译上一次选择的内核的编译命令，使用下面的命令无需通过图形界面选择，可以直接开始编译内核源码

[o.k.] Repeat Build Options [**sudo ./build.sh BOARD=orangepi3 BRANCH=legacy BUILD_OPT=kernel KERNEL_CONFIGURE=yes**]

7) 查看编译生成的内核相关的 deb 包

a. **linux-dtb-legacy-sun50iw6_2.1.0_arm64.deb** 包含内核使用的 dtb 文件

b. **linux-headers-legacy-sun50iw6_2.1.0_arm64.deb** 包含内核头文件

c. **linux-image-legacy-sun50iw6_2.1.0_arm64.deb** 包含内核镜像和内核模块

```
test@test:~/orangepi-build$ ls output/debs/linux-*
output/debs/linux-dtb-legacy-sun50iw6_2.1.0_arm64.deb
output/debs/linux-headers-legacy-sun50iw6_2.1.0_arm64.deb
output/debs/linux-image-legacy-sun50iw6_2.1.0_arm64.deb
```

8) 生成的 linux-image 的 deb 包包含的文件如下所示

a. 使用下面的命令可以解压 deb 包

```
test@test:~/orangepi-build$ cd output/debs
test@test:~/orangepi_build/output/debs$ mkdir test
test@test:~/orangepi_build/output/debs$ cp \
linux-image-legacy-sun50iw6_2.1.0_arm64.deb test/
test@test:~/orangepi_build/output/debs$ cd test
test@test:~/orangepi_build/output/debs/test$ dpkg -x \
linux-image-legacy-sun50iw6_2.1.0_arm64.deb .
test@test:~/orangepi_build/output/debs/test$ ls
boot    etc    lib    linux-image-legacy-sun50iw6_2.1.0_arm64.deb    usr
```

b. 解压后的文件如下所示

```
test@test:~/orangepi_build/output/debs/test$ tree -L 2
.
├── boot
│   ├── config-4.9.118-sun50iw6          //编译内核源码使用的配置文件
│   ├── System.map-4.9.118-sun50iw6
│   └── vmlinuz-4.9.118-sun50iw6         //编译生成的内核镜像文件
├── etc
│   └── kernel
├── lib
│   └── modules                          //编译生成的内核模块
├── linux-image-legacy-sun50iw6_2.1.0_arm64.deb
└── usr
    ├── lib
    └── share

8 directories, 4 files
```

9) orangepi-bulid 编译系统编译 linux 内核源码时首先会将 linux 内核源码和 github 服务器的 linux 内核源码进行同步，所以如果想修改 linux 内核的源码，首先需要关闭源码的更新功能（**需要完整编译过一次 linux 内核源码后才能关闭这个功能，否则会提示找不到 linux 内核的源码**），否则所作的修改都会被还原，方法如下：

设置 userpatches/config-default.conf 中的 IGNORE_UPDATES 变量为“yes”

```
test@test:~/orangepi-build$ vim userpatches/config-default.conf
IGNORE_UPDATES="yes"
```

10) 如果对内核做了修改，可以使用下面的方法来更新开发板 linux 系统的内核和内核模块

a. 将编译好的 linux 内核的 deb 包上传到开发板的 linux 系统中

```
test@test:~/orangepi-build$ cd output/debs
test@test:~/orangepi-build/output/debs$ scp \
linux-image-legacy-sun50iw6_2.1.0_arm64.deb root@192.168.1.207:/root
```

b. 然后登录到开发板，卸载已安装的 linux 内核的 deb 包

```
root@orangepi:~# apt purge -y linux-image-legacy-sun50iw6
```

c. 再安装刚才上传的新的 linux 内核的 deb 包

```
root@orangepi:~# dpkg -i linux-image-legacy-sun50iw6_2.1.0_arm64.deb
```

d. 然后重启开发板，再查看内核相关的修改是否已生效

11) 安装内核头文件到 linux 系统中的方法如下所示

a. 将编译好的 linux 头文件的 deb 包上传到开发板的 linux 系统中

```
test@test:~/orangepi-build$ cd output/debs
test@test:~/orangepi-build/output/debs$ scp \
linux-headers-legacy-sun50iw6_2.1.0_arm64.deb root@192.168.1.207:/root
```

b. 然后登录到开发板，安装刚才上传的的 linux 头文件的 deb 包

```
root@orangepi:~# dpkg -i linux-headers-legacy-sun50iw6_2.1.0_arm64.deb
```

c. 安装完后在/usr/src 中就可以看到刚才安装的内核头文件相关的内容了

```
root@orangepi:~# ls /usr/src
linux-headers-4.9.118-sun50iw6
root@orangepi:~# ls /usr/src/linux-headers-4.9.118-sun50iw6
Documentation   Module.symvers   certs   firmware   init   lib   net   security   usr
Kconfig   arch   crypto   fs   ipc   mm   samples   sound   virt   Makefile   block
drivers   include   kernel   modules   scripts   tools
```

4.4. 编译 rootfs

1) 运行 build.sh 脚本，记得加 sudo 权限

```
test@test:~/orangepi-build$ sudo ./build.sh
```

2) 选择 **Rootfs and all deb packages**，然后回车

```
Choose an option
Compile image | rootfs | kernel | u-boot

        U-boot package
        Kernel package
        Rootfs and all deb packages
        Full OS image for flashing
```

3) 接着选择开发板的型号

```
Choose an option
Please choose a Board.

orangepir1            Allwinner H2+ quad core 256MB RAM WiFi SPI 2xETH
orangepizero          Allwinner H2+ quad core 256MB/512MB RAM WiFi SPI
orangepipc            Allwinner H3 quad core 1GB RAM
orangepipcplus        Allwinner H3 quad core 1GB RAM WiFi eMMC
orangepione           Allwinner H3 quad core 512MB RAM
orangepilite          Allwinner H3 quad core 512MB RAM WiFi
orangepiplus          Allwinner H3 quad core 1GB/2GB RAM WiFi GBE eMMC
orangepiplus2e        Allwinner H3 quad core 2GB RAM WiFi GBE eMMC
orangepizeroplus2h3   Allwinner H3 quad core 512MB RAM WiFi/BT eMMC
orangepipc2           Allwinner H5 quad core 1GB RAM GBE SPI
orangepiprime         Allwinner H5 quad core 2GB RAM GBE WiFi/BT
orangepizeroplus      Allwinner H5 quad core 512MB RAM GBE WiFi SPI
orangepizeroplus2h5   Allwinner H5 quad core 512MB RAM WiFi/BT eMMC
orangepi3             Allwinner H6 quad core 1GB/2GB RAM GBE WiFi/BT eMMC USB3
orangepilite2         Allwinner H6 quad core 1GB RAM WiFi/BT USB3
orangepioneplus       Allwinner H6 quad core 1GB RAM GBE
orangepizero2         Allwinner H616 quad core 512MB/1GB RAM WiFi/BT GBE SPI
```

4) 然后选择 rootfs 的类型

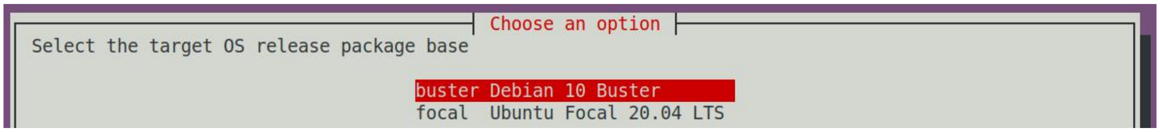
buster	Debian 10
bionic	Ubuntu 18.04
focal	buntu 20.04

a. linux4.9 支持的 linux 发行版如下所示

```
Choose an option
Select the target OS release package base

        buster Debian 10 Buster
        bionic Ubuntu Bionic 18.04 LTS
        focal  Ubuntu Focal 20.04 LTS
```

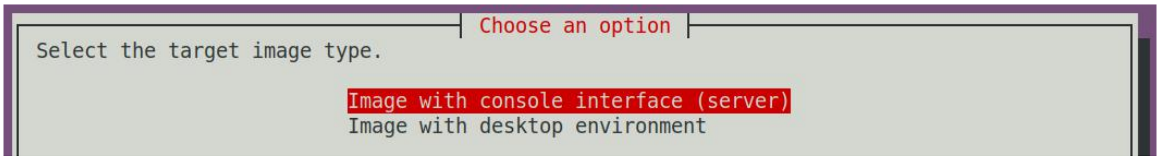
b. linxu5.4 支持的 linux 发行版如下所示



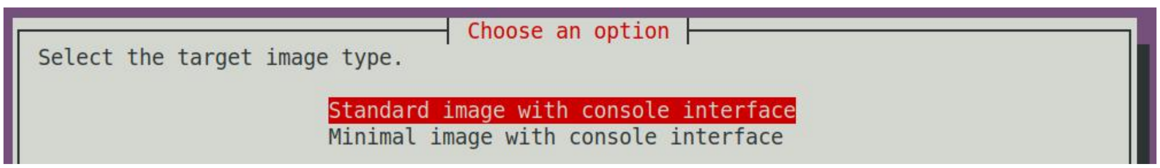
5) 然后选择镜像的类型

a. **Image with console interface (server)**表示服务器版的镜像，体积比较小

b. **Image with desktop environment** 表示带桌面的镜像，体积比较大



6) 如果是编译服务器版的镜像，还可以选择编译 Standard 版本或者 Minimal 版本，Minimal 版本预装的软件会比 Standard 版本少很多



7) 选择镜像的类型后就会开始编译 rootfs，编译时提示的部分信息说明如下

a. rootfs 的类型

[o.k.] local not found [Creating new rootfs cache for **focal**]

b. 编译生成的 rootfs 压缩包的存放路径

[o.k.] Target directory [**external/cache/rootfs**]

c. 编译生成的 rootfs 压缩包的名字

[o.k.] File name [**focal-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4**]

d. 编译使用的时间

[o.k.] Runtime [**13 min**]

e. 重复编译 rootfs 的命令，使用下面的命令无需通过图形界面选择，可以直接开始编译 rootfs

[o.k.] Repeat Build Options [**sudo ./build.sh BOARD=orangepi3 BRANCH=legacy BUILD_OPT=rootfs RELEASE=focal BUILD_MINIMAL=no BUILD_DESKTOP=no KERNEL_CONFIGURE=yes**]

8) 查看编译生成的 rootfs 压缩包

a. focal-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4 是 rootfs 的压缩包，名字各字段的含义为

   a) **focal** 表示 rootfs 的 linux 发行版的类型

   b) **cli** 表示 rootfs 为服务器版的类型，如果为 **dekstop** 则表示桌面版类型

   c) **arm64** 表示 rootfs 的架构类型

   d) **153618961f14c28107ca023429aa0eb9** 是由 rootfs 安装的所有软件包的包名生成的 MD5 哈希值，只要没有修改 rootfs 安装的软件包的列表，那么这个值就不会变，编译脚本会通过这个 MD5 哈希值来判断是否需要重新编译 rootfs

b. focal-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4.list 列出了 rootfs 安装的所有软件包的包名

```
test@test:~/orangepi-build$ ls external/cache/rootfs/
focal-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4
focal-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4.list
```

9) 如果需要的 rootfs 在 **external/cache/rootfs** 下已经存在，那么再次编译 rootfs 就会直接跳过编译过程，不会重新开始编译，编译镜像的时候也会去 **external/cache/rootfs** 下查找是否已经有缓存可用的 rootfs，如果有就直接使用，这样可以节省大量的下载编译时间

10) 由于编译 rootfs 的时间较长，如果不想从头开始编译 rootfs，或者编译 rootfs 的过程有问题，可以直接下载 Orange Pi 缓存的 rootfs 压缩包，rootfs 压缩包百度云盘的下载链接如下所示，下载好的 rootfs 压缩包（不要解压哦）需要放在 orangepi-build 的 **external/cache/rootfs** 目录下才能被编译脚本正常使用

```
链接：https://pan.baidu.com/s/1vWQmCmSYdH7iCDFyKpJtVw
提取码：zero
```

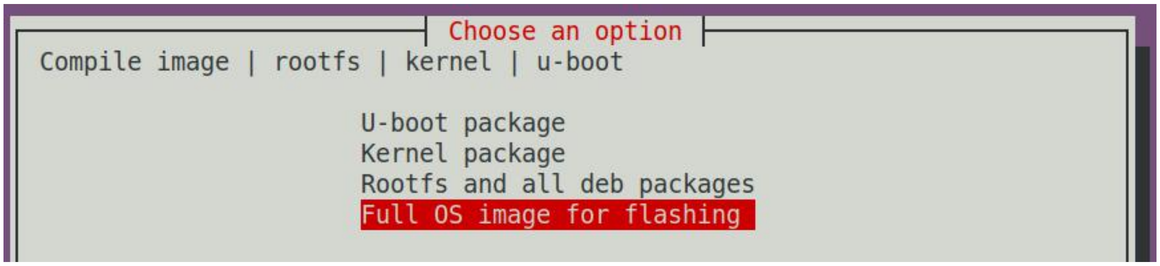


4.5. 编译 linux 镜像

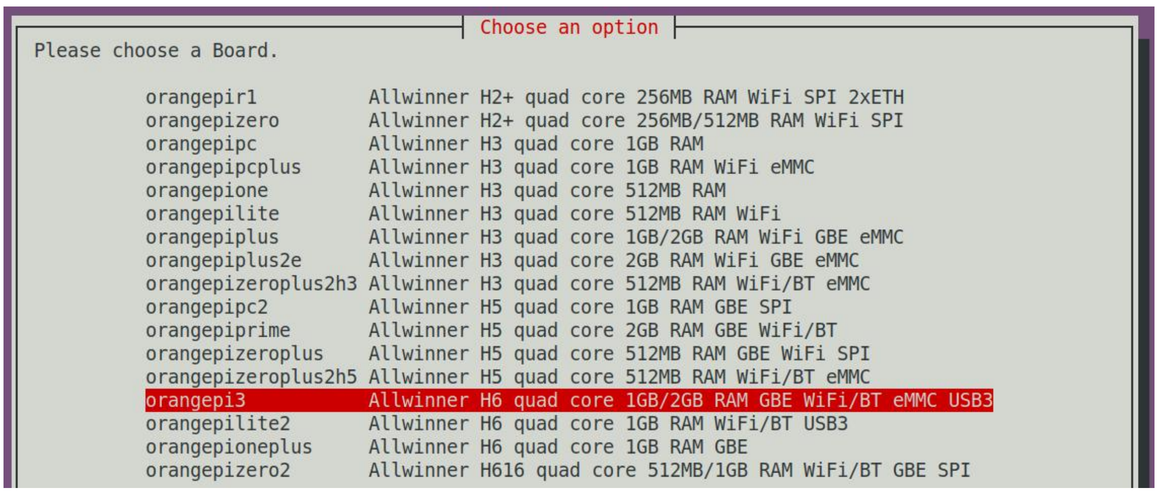
1) 运行 build.sh 脚本，记得加 sudo 权限

```
test@test:~/orangepi-build$ sudo ./build.sh
```

2) 选择 **Full OS image for flashing**，然后回车



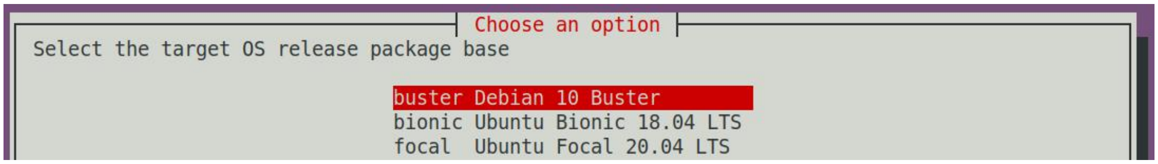
3) 然后选择开发板的型号



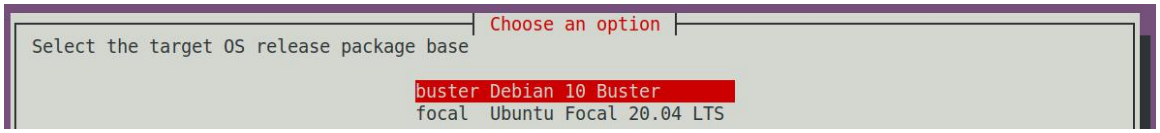
4) 然后选择 rootfs 的类型

buster	Debian 10
bionic	Ubuntu 18.04
focal	buntu 20.04

a. linux4.9 支持的 linux 发行版如下所示



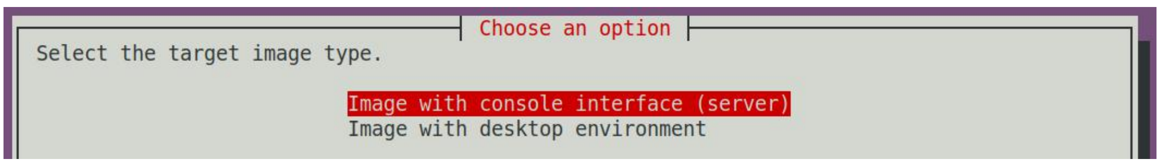
b. linxu5.4 支持的 linux 发行版如下所示



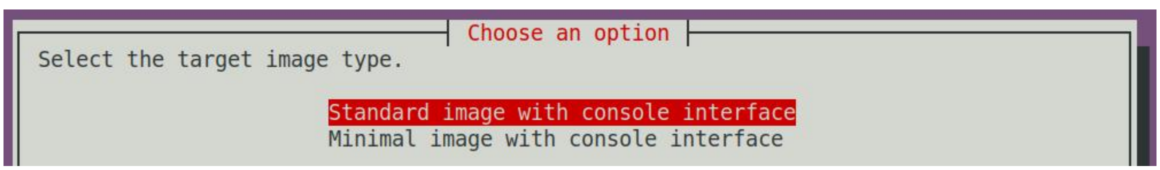
5) 然后选择镜像的类型

a. **Image with console interface (server)**表示服务器版的镜像，体积比较小

b. **Image with desktop environment** 表示带桌面的镜像，体积比较大



6) 如果是编译服务器版的镜像，还可以选择编译 Standard 版本或者 Minimal 版本，Minimal 版本预装的软件会比 Standard 版本少很多



7) 选择镜像的类型后就会开始编译 linux 镜像，编译的大致流程如下

a. 初始化 Ubuntu PC 的编译环境，安装编译过程需要的软件包

b. 下载 u-boot 和 linux 内核的源码（如果已经缓存，则只更新代码）

c. 编译 u-boot 源码，生成 u-boot 的 deb 包

d. 编译 linux 源码，生成 linux 相关的 deb 包

e. 制作 linux firmware 的 deb 包

f. 制作 orangepi-config 工具的 deb 包

g. 制作板级支持的 deb 包

h. 如果是编译 desktop 版镜像，还会制作 desktop 相关的 deb 包

i. 检查 rootfs 是否已经缓存，如果没有缓存，则重新制作 rootfs，如果已经缓存，则直接解压使用

j. 安装前面生成的 deb 包到 rootfs 中

k. 对不同的开发板和不同类型镜像做一些特定的设置，如预装额外的软件包，修改系统配置等
l. 然后制作镜像文件，并格式化分区，默认类型为 ext4
m. 再将配置好的 rootfs 拷贝到镜像的分区中
n. 然后更新 initramfs
o. 最后将 u-boot 的 bin 文件通过 dd 命令写入到镜像中

8) 编译完镜像后会提示下面的信息（以 legacy 分支为例）
a. 编译生成的镜像的存放路径

```
[ o.k. ] Done building
[ output/images/orangepi3_2.1.0_ubuntu_focal_server_linux4.9.118/orangepi3_2.1.0_ubuntu_focal_server_linux4.9.118.img ]
```

b. 编译使用的时间

```
[ o.k. ] Runtime [ 19 min ]
```

c. 重复编译镜像的命令，使用下面的命令无需通过图形界面选择，可以直接开始编译镜像

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh  BOARD=orangepi3 BRANCH=legacy BUILD_OPT=image RELEASE=focal  BUILD_MINIMAL=no BUILD_DESKTOP=no KERNEL_CONFIGURE=yes ]
```

5. Android 系统使用说明

5. 1. 已支持的 Android 版本

Android 版本	内核版本
Android 7.0	linux3.10

5. 2. Android 7.0 功能适配情况

功能	状态
HDMI 视频	OK
HDMI 音频	OK
USB2.0	OK

USB3.0 x 4	OK
TF 卡启动	OK
EMMC 启动	OK
网卡	OK
红外	OK
WIFI	OK
WIFI hotsport	OK
蓝牙	OK
蓝牙耳机	OK
BLE 蓝牙	OK
耳机音频	OK
CVBS 视频	OK
USB 摄像头	OK
LED 灯	OK
温度传感器	OK
Mali GPU	OK
视频编解码	OK
电源按键	OK
ADB 调试	OK
mini PCIE	NO

5.3. 板载 LED 灯显示说明

	黄灯	红灯
u-boot 启动阶段	灭	亮
内核启动到进入系统	亮	灭
GPIO 口	PL7	PL4

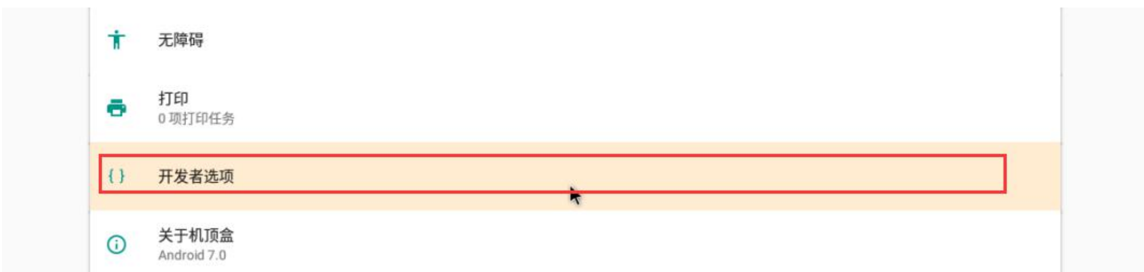
5.4. ADB 的使用方法

5.4.1. 打开 USB debugging 选项

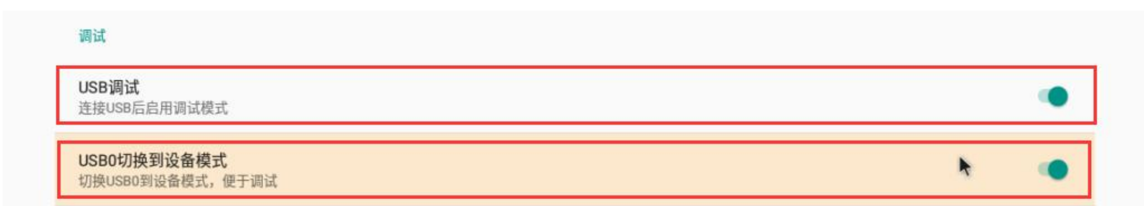
1) USB 调试模式默认是打开的，ADB 调试默认是可以直接使用的，如果 ADB 使用有问题，可以使用下面的方式打开，首先选择设置



2) 然后选择**开发者选项**



3) 然后打开 **USB 调试**和 **USB 切换到设备模式**



5.4.2. 使用网络连接 adb 调试

1) 使用网络 adb 无需使用 Micro USB 接口的数据线来连接电脑和开发板，而是通过网络来通信，所以首先请确保开发板的有线或者无线网络已经连接好了，然后获取开发板的 IP 地址，后面要用到

2) 确保已经打开 **USB debugging** 选项

3) 确保 Android 系统的 **service.adb.tcp.port** 设置为 5555 端口号

```
petrel-p1:/ # getprop | grep "adb.tcp"
[service.adb.tcp.port]: [5555]
```

4) 如果 **service.adb.tcp.port** 没有设置，可以使用下面的命令设置网络 adb 的端口号

```
petrel-p1:/ # setprop service.adb.tcp.port 5555
petrel-p1:/ # stop adbd
petrel-p1:/ # start adbd
```

5) 在 Ubuntu PC 上安装 adb 工具

```
test@test:~$ sudo apt update
test@test:~$ sudo apt install adb
```

6) 然后在 Ubuntu PC 上连接网络 adb

```
test@test:~$ adb connect 192.168.1.xxx     (IP 地址需要修改为开发板的 IP 地址)
* daemon not running; starting now at tcp:5037
* daemon started successfully
connected to 192.168.1.xxx:5555

test@test:~$ adb devices
List of devices attached
192.168.1.xxx:5555          device
```

7) 然后在 Ubuntu PC 上通过 adb shell 就可以登录 android 系统

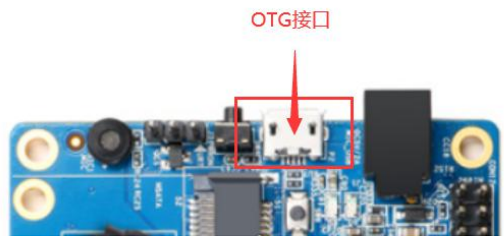
```
test@test:~$ adb shell
petrel-p1:/ #
```

5.4.3. 使用数据线连接 adb 调试

1) 首先确保打开 **USB debugging** 选项

2) 首先需要使用 Micro USB 接口的数据线将开发板连接到电脑的 USB 接口中（请同时使用 DC 电源给开发板供电）





3) 在 Ubuntu PC 上安装 adb 工具

```
test@test:~$ sudo apt update
test@test:~$ sudo apt install adb
```

4) 查看识别到 ADB 设备

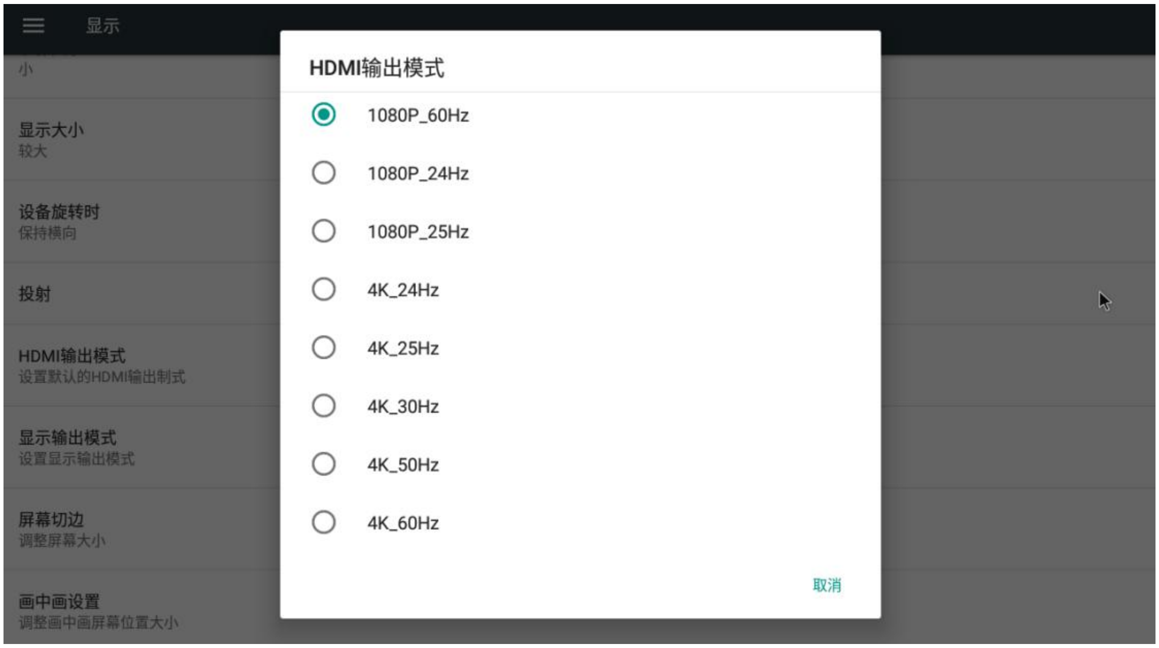
```
test@test:~$ adb devices
List of devices attached
20080411     device
```

5) 然后在 Ubuntu PC 上通过 adb shell 就可以登录 android 系统

```
test@test:~$ adb shell
petrel-p1:/ #
```

5.5. HDMI 4K 显示说明

1) 如果开发板的 HDMI 接到不支持 4K 的电视上或者显示器上，在设置中虽然能查看到 HDMI 支持的分辨率有 4K 相关的选项，但是是无法使用的



在 1080p 的电视中选择 4K 选项会提示 HDMI 输出模式不支持



2) 只有将开发板的 HDMI 接到支持 4K 的电视上或者显示器上，HDMI 4K 相关的分辨率选项才能正常使用

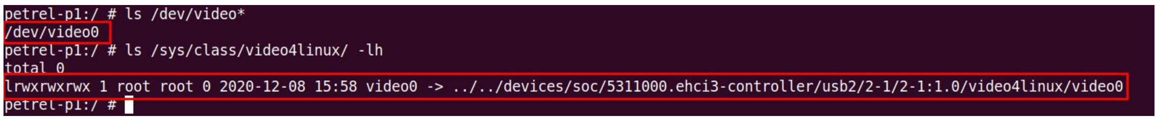
5. 6. USB 摄像头使用方法

1) 先在开发板的 USB 接口中插入 USB 摄像头，然后确认下 USB 摄像头相关的内核模块已正常加载

```
petrel-p1:/ # lsmod
Module                  Size    Used by
bcmdhd                943519    0
```

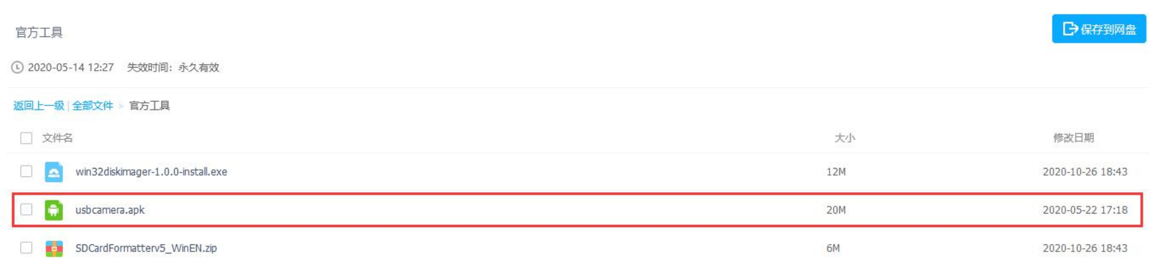
```
bcm_btlpm_rc              7412   0
uvcvideo                 68460   0
videobuf2_vmalloc         3263   1 uvcvideo
videobuf2_memops          2691   1 videobuf2_vmalloc
videobuf2_core           31877   1 uvcvideo
```

2) USB 摄像头如果识别正常，在/dev 下会生成相应的 video 设备节点



3) 然后确保 Ubuntu PC 和开发板的 adb 连接正常

4) 在 Orange Pi 3 的资料下载页面的**官方工具**中下载 USB 摄像头测试 APP



5) 然后使用 adb 命令安装 USB 摄像头测试 APP 到 Android 系统中，当然也可以使用 U 盘拷贝的方式进行安装

```
test@test:~$ adb install usbcamera.apk
```

6) 安装完后在 Android 的桌面中可以看到 USB 摄像头的启动图标

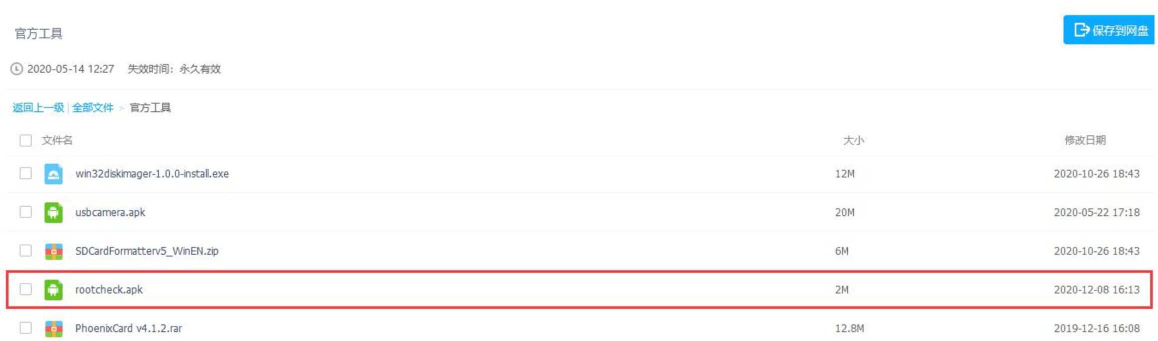


7) 然后双击打开 USB 摄像头 APP 就可以看到 USB 摄像头的输出视频了

5.7. Android 系统 ROOT 说明

Orange Pi 发布的 Android 7.0 系统已经 ROOT，可以使用下面的方法来测试

1) 在 Orange Pi 3 资料下载页面的**官方工具**中下载 rootcheck.apk



2) 然后确保 Ubuntu PC 和开发板的 adb 连接正常

3) 然后使用 adb 命令安装 rootcheck.apk 到 Android 系统中，当然也可以使用 U 盘拷贝的方式进行安装

```
test@test:~$ adb install rootcheck.apk
```

4) 安装完后在 Android 的桌面可以看到 ROOT 测试工具的启动图标



5) 第一次打开 **ROOT 测试工具**后的显示界面如下图所示



6) 然后就可以点击“**立刻检查**”开始 Android 系统的 ROOT 状态的检查，检查完后的显示如下所示，可以看到 Android 系统已取得 ROOT 权限



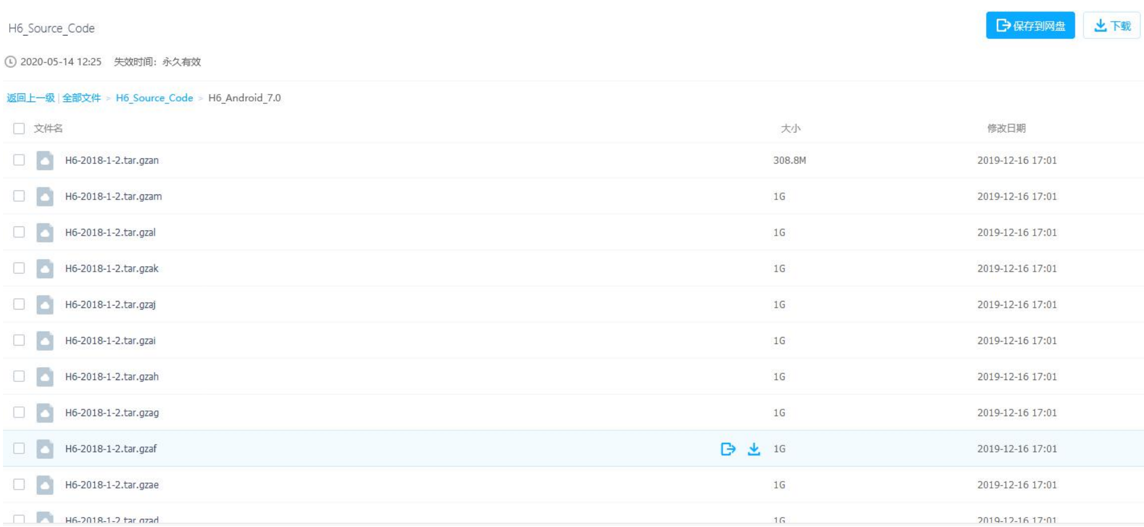
6. Android SDK 使用说明

1) Android SDK 的编译是在安装有 **Ubuntu 14.04** 的 PC 上进行的，其它版本的 Ubuntu 系统可能会有一些区别

2) Android SDK 是芯片厂商释放的原始的 SDK，如果想在 Orange Pi 开发板上使用 Android SDK 编出来的 Android 镜像，**需要针对不同的板子进行适配，才能保证所有功能使用正常**

6.1. 下载 Android SDK 的源码

1) 首先从百度云盘下载 Android SDK 的分卷压缩包



2) Android 源码包下载完后，首先需要将多个压缩文件合并成一个，然后解压

```
test@test:~$ mkdir OrangePiH6
test@test:~$ cat H6-2018-1-2.tar.gza* > OrangePiH6.tar
test@test:~$ tar xf OrangePiH6.tar -C OrangePiH6
```

6.2. 搭建 Android 编译环境

1) 安装 JDK

```
test@test:~$ sudo add-apt-repository ppa:openjdk-r/ppa
test@test:~$ sudo apt-get update
```

```
test@test:~$ sudo apt-get install openjdk-8-jdk
```

2) 配置 JAVA 环境变量

a. 首先确定 java 的安装路径，一般为

```
test@test:~$ ls /usr/lib/jvm/java-8-openjdk-amd64
ASSEMBLY_EXCEPTION  bin  docs  include  jre  lib  man  src.zip
THIRD_PARTY_README
```

b. 然后使用下面的命令导出 java 的环境变量

```
test@test:~$ export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
test@test:~$ export PATH=$JAVA_HOME/bin:$PATH
test@test:~$ export CLASSPATH=.:$JAVA_HOME/lib:$JAVA_HOME/lib/tools.jar
```

3) 安装平台支持软件

```
test@test:~$ sudo apt-get update
test@test:~$ sudo apt-get install git gnupg flex bison gperf build-essential \
zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 \
lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z1-dev ccache \
libgl1-mesa-dev libxml2-utils xsltproc unzip

test@test:~$ sudo apt-get install u-boot-tools
```

6.3. 编译 Android 镜像

6.3.1. 编译内核

1) Android SDK解压后，在解压的目录下会有andorid和lichee两个子目录，lichee目录主要内容如下所示

```
lichee/brandy/u-boot-2014.07              #uboot代码目录
lichee/bootloader/uboot_2014_sunxi_spl    #boot0 代码目录
lichee/linux-3.10                         #内核代码
lichee/tools                              #方案硬件配置，打包工具等
```

2) lichee的编译方法如下所示

```
test@test:~$ cd OrangePiH6/lichee
test@test:~/OrangePiH6/lichee$ ./build.sh config
```

```
Welcome to mkscript setup progress
All available chips:
    0. sun50iw1p1
    1. sun50iw2p1
    2. sun50iw6p1
    3. sun8iw11p1
    4. sun8iw12p1
    5. sun8iw6p1
    6. sun8iw7p1
    7. sun8iw8p1
    8. sun9iw1p1
Choice: 2
All available platforms:
    0. android
    1. dragonboard
    2. linux
    3. eyeseelinux
Choice: 0
All available business:
    0. 5.1
    1. 4.4
    2. 7.x
Choice: 2
```

3) 编译成功后输出内容如下所示

```
sun50iw6p1 compile Kernel successful

INFO: build kernel OK.

INFO: build rootfs ...
INFO: skip make rootfs for android
INFO: build rootfs OK.
----------------------------------------
build sun50iw6p1 android 7.x lichee OK
```

```
----------------------------------------
```

6.3.2. 编译 Android 源码

1) 编译 Android 的命令如下

```
test@test:~/OrangePiH6$ cd android
test@test:~/OrangePiH6/android$ source build/envsetup.sh
test@test:~/OrangePiH6/android$ lunch petrel_fvd_p1-eng
test@test:~/OrangePiH6/android$ extract-bsp
test@test:~/OrangePiH6/android$ make -j8
test@test:~/OrangePiH6/android$ pack
```

2) 其中 pack 命令用于打包生成安卓固件，如果编译打包过程都顺利通过，会提示下面的信息

```
Dragon execute image.cfg SUCCESS !
----------image is at----------

OrangePiH6/lichee/tools/pack/sun50iw6p1_android_petrel-p1_uart0.img

pack finish
```

3) 生成的安卓镜像存放的路径为

```
OrangePiH6/lichee/tools/pack/sun50iw6p1_android_petrel-p1_uart0.img
```